Б. А. Романов
А. С. Кушниренко

dBASE IV

Назначение, функции, применение

Москва
«Радио и связь»
1991

УДК 681.3

Романов Б.А., Кушниренко А.С. dBASE IV. Назначение, функции, применение. - М.: Радио и связь, 1991. - 384 с.: ил. - **ISBN 5-256-00822-6.**

В книге на примере широко известного семейства dBASE излагается назначение, принципы построения, функции и опыт практического использования системы управления базами данных (СУБД) реляционного типа. Большое внимание уделяется описанию структуры и способов построения интерфейса пользователя, управляемого с помощью меню. Рассмотрена концепция, синтаксис и функции языков высокого уровня dBASE и SQL, применяемых для программирования информационных задач. Изложение материала сопровождается многочисленными примерами. В частности, работа с управляющим центром пакета dBASE IV, который реализует управляемый с помощью меню интерфейс пользователя, поясняется на примере задачи учета перевозок автопредприятием. Приводится библиотечка процедур, которые можно использовать в виде стандартных подпрограмм для выполнения часто повторяющихся операций. Подробно излагаются принципы и методы разработки прикладных программ на языке dBASE IV. Дается описание прикладной программы Service, обеспечивающей выполнение сервисных функций по обработке файлов БД. Примеры и прикладные программы, приводимые в книге, записаны на дискету, которая прилагается.

Для научных работников, специализирующихся в области создания программных продуктов; может быть полезна широкому кругу пользователей.

Табл. 21. Ил. 13.

Рецензент: Т.Э. Кренкель

Редакция литературы по информатике и вычислительной технике

Научное издание

Романов Борис Александрович, **Кушниренко** Александр Сергеевич

dBASE IV. Назначение, функции, применение

Заведующая редакцией Г.И. Козырева
Редактор В.А. Солдатенков
Переплет художника В.Е. Самохина
Художественный редактор А.В. Проценко
Технический редактор С.Ф. Романова
Корректор М.Д. Шунина

ИБ N 2445

Подписано в печать с оригинал-макета 28. 09. 90. Формат 60×88 1/16. Бумага офс. № 1. Гарнитура журн.-рубл. Печать офсетная. Усл.печ.л. 23,52. Усл.кр.-отт. 23,89. Уч.изд.л. 24,90. Тираж 30 000 экз. Изд. № 23393. Зак. № 6340. Цена 4 р. 50 к.

Издательство "Радио и связь", 101000 Москва, Почтамт, а/я 693

Ордена Октябрьской Революции и ордена Трудового Красного Знамени МПО "Первая Образцовая типография" Государственного комитета СССР по печати.
113054, Москва, Валовая, 28

$$Р\,\frac{2404090000-097}{046(01)-91}\,КБ-37-21-90$$

ISBN 5-256-00822-6

ПРЕДИСЛОВИЕ

Доступность информации, объем и скорость ее обработки становятся решающими факторами развития производительных сил государства, науки, культуры, общественных институтов и т.д. Переработка информации вплоть до конца 70-х годов осуществлялась в основном на больших и средних ЭВМ. В них создавались большие банки данных, а ввод и вывод информации осуществлялся операторами с удаленных терминалов. По существу, весь процесс переработки информации выполнялся профессионалами, и конечный пользователь чаще всего не имел к ней быстрого доступа. Разработка и бурное распространение с конца 70-х годов мини- и микрокомпьютеров, а также персональных компьютеров (ПК) не только приблизили конечного, часто профессионально не подготовленного пользователя к процессу переработки информации, но и позволили активно в него включиться.

Большую роль в этом сыграли специально разработанные для ПК программные средства - текстовые редакторы, системы управления базами данных (СУБД), системы автоматизации проектных работ и т.д. Их отличительная особенность от программных продуктов для больших и средних ЭВМ заключается в создании более удобного - "дружественного" интерфейса пользователя. Соединение ПК с большими и средними ЭВМ предоставляет конечному пользователю их информационные ресурсы и оставляет все преимущества ПК.

Среди СУБД для ПК наиболее распространенным в мире является семейство dBASE фирмы Ashton-Tate. Такая популярность объясняется отчасти исторически, так как эта СУБД появилась одной из первых на рынке программных продуктов данного профиля. Но не в меньшей степени этому способствовала удачность заложенных основ конструкции СУБД и командного языка. Предыдущая версия этого семейства dBASE III+ широко используется в СССР. По ней издано несколько книг, например Д.Г.Берещанский "Практическое программирование на dBASE", Р.Крамм "Системы управления базами данных dBASE II и dBASE III для персональных компьютеров". В новой версии dBASE IV концепция СУБД и командный язык получили значительное развитие, позволившее устранить недостатки предыдущей версии, а также существенно обогатить палитру средств обработки данных и улучшить интерфейс пользователя. В данной книге выполнен анализ структуры и компонентов СУБД и командного языка версии 1.0 dBASE IV и на многочисленных примерах показаны их функции и возможности.

Главы 1, 3 и 4 написаны Б.А. Романовым, главы 2 и 5 - совместно Б.А. Романовым и А.С. Кушниренко.

В книге упоминается учебная дискета, сведения о которой читатель найдет в рекламном объявлении на 2-й стр. обложки.

Глава 1
ОБЩИЕ СВЕДЕНИЯ О ПАКЕТЕ dBASE IV

1.1. СРАВНИТЕЛЬНЫЕ ХАРАКТЕРИСТИКИ СУБД РЕЛЯЦИОННОГО ТИПА

Информация в ЭВМ записывается в закодированном виде на различных носителях (чаще всего используются магнитные поверхности - диски, ленты и др.). В настоящее время наиболее распространена система кодирования, в которой каждый символ информации представляется в виде двоичного числа из 8 разрядов. Единицей количества информациии служит бит (один двоичный разряд) или байт (8 бит).

Хранение информации осуществляется в виде записей, составленных из блоков данных различного типа (чисел, символов и т.д.), относящихся к одному информационному объекту. Примером записи служит совокупность данных в библиографической карточке. Информация определенного профиля, отделяемая в записи специальными разделителями, называется полем записи (например, в библиографической карточке полями являются название книги, фамилия автора, год издания и т.д.). Совокупность записей составляет файл данных, а набор файлов образует базу данных (БД). Операции по обеспечению хранения и обработки данных в файлах выполняются с помощью операционной системы (ОС) ПК. Для более удобного обращения к файлам на дисках в развитой ОС создается хранилище файлов - файловая система, в которой файлы объединяются в группы по их целевому назначению. Каждой группе присваивается имя - директорий этой группы. Совокупность директориев рассматривается в ОС как специальный файл, которому также присваивается имя - директорий более высокого уровня. Так образуется древовидная иерархическая структура директориев.

Директорий самого высокого уровня называется корневым, а на самом нижнем уровне директориев находятся файлы пользователей. Для обращения к файлу задается маршрут по директориям, который включает имя файла и перечень директориев (от корневого до ведущих к искомому файлу) по следующей форме: **C:\MAIN\BASA\PROFIL\Detail**, где **C:** - имя диска; первая наклонная черта после имени диска означает корневой директорий, а остальные служат разделителями между именами директориев; **MAIN**, **BASA** и **PROFIL** - директории нижестоящих уровней; **Detail** - имя файла. Заметим, что имена дисков, файлов

и директориев можно записывать как строчными, так и прописными буквами.

Для того чтобы более эффективно использовать несколько ПК, выполняющих обработку общей информации, создаются локальные вычислительные сети ПК (ЛВС). Организация ЛВС позволяет использовать общие ресурсы ПК, передаваемые в состав ЛВС, периферийные устройства - дисковые устройства большой емкости, принтеры, плоттеры, сканеры и т.д., а также программы и данные. В ЛВС, по крайней мере, один ПК большой производительности выполняет роль так называемого файлового сервера, управляющего передачей данных и разделением периферийных устройств. Остальные ПК называются рабочими станциями или узлами ЛВС.

Типы рабочих станций определяются функциями ЛВС. Например, в состав ЛВС могут входить маломощные ПК с небольшой оперативной памятью и невысоким быстродействием, без гибких и жестких дисков, предназначенные для ввода данных с клавиатуры и записи ее на периферийный диск большой емкости, в котором размещается банк данных. Другие ПК в составе ЛВС могут иметь различные конфигурации внешних устройств, в которых варьируется количество гибких и жестких дисков, принтеров и т.д. Например, могут быть конфигурации с одним или двумя гибкими дисками без жесткого или с жестким диском, с принтером или без него и т.д. в зависимости от выполняемых функций. Разделенные прикладные программы загружаются в рабочую станцию из файлового сервера и выполняются в ней, а разделенные данные передаются из файлового сервера по сети в рабочую станцию и в ней обрабатываются.

Операционная система служит для общей организации процесса работы ПК и обеспечения простейших операций по обработке данных. Для того чтобы выполнить более сложные операции по манипулированию данными на ПК (поиск и отбор данных, обеспечение удобного ввода и обновления данных, формирование и печать выходных документов и т.д.) разрабатываются различные системы управления базами данных (СУБД).

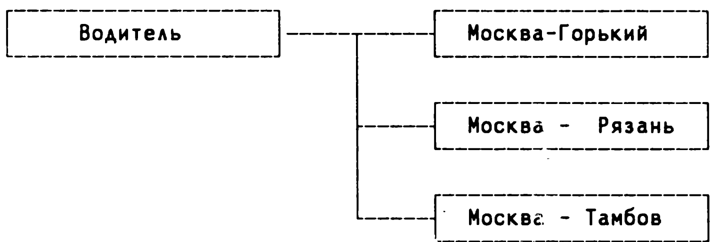
При решении многих информационных задач в настоящее время широко используются СУБД реляционного типа. В этих СУБД реализуется реляционная модель данных - представление их в табличном виде. Строка такой таблицы эквивалентна записи файла БД, а колонка - полю записи. Доступ к элементу данных осуществляется посредством связи требуемой строки (записи) с требуемой колонкой (полем). Многие СУБД реляционного типа позволяют языковыми средствами поддерживать и другие модели данных, например, иерархическую. В иерархической модели данные связываются по уровням. Данные высшего уровня называются родительскими, а связанные с ними данные следующего уровня - сыновними. Иерархическая модель

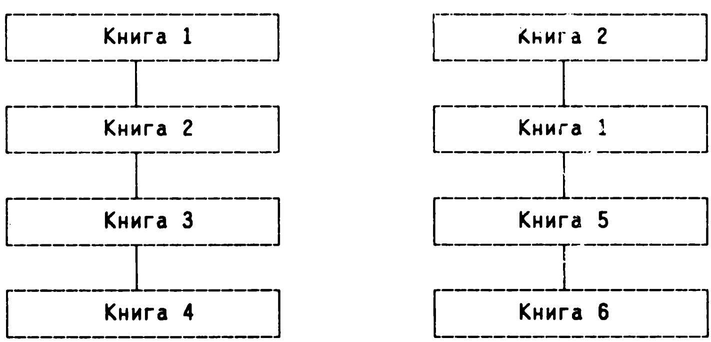
формируется за счет связи таблиц по общему полю, отдельной записи и т.п.

Можно выделить три основных типа связей записей: одна-одна запись; одна-несколько (несколько-одна) записей; несколько-несколько записей.

Примером связи записей одна-одна служит запись, включающая фамилию, имя и отчество и запись биографии одного человека. Связь типа одна-несколько записей - это, например, записи, включающие данные о рейсах, выполненных водителем за определенный период времени. Связь типа несколько-несколько записей можно представить в виде перекрестных ссылок, которые имеются в книгах.







К числу СУБД реляционного типа, предназначенных для ПК, относятся: семейство dBASE (dBASE II, III, III+ и IV), Clipper, Foxbase, R:BASE, Paradox и др. В этих СУБД записи и соответственно поля имеют обычно фиксированную длину (чаще всего длина такой записи достигает 4000...5000 байт). Исключение составляют поля типа Memo, используемые, например, в СУБД

семейства dBASE. Число полей в перечисленных СУБД варьируется от 128 до 1024.

Длина поля зависит от типа поля и может составлять от 255 до 4000 байт для текстовых полей, до 20 байт для числовых полей и имеет фиксированные значения для полей типа даты (8 байт) и логических полей (1 байт). Поле типа Memo служит для хранения больших массивов текстовой информации и хранится в отдельном файле БД, но воспринимается как поле в составе основного файла БД. Это поле имеет плавающую длину, определяемую объемом введенной текстовой информации, и может достигать 32767 байт.

Большинство СУБД реляционного типа для ПК позволяют создавать файлы с числом записей до 1 млрд. и объемом до 2 тыс. Мбайт. Обычно ограничением на число и объем записей в первую очередь выступает емкость внешней памяти ПК. В настоящее время в широкой практике используются твердые диски (с магнитной запоминающей поверхностью) с емкостью памяти до 100...300 Мбайт.

В состав многих СУБД, предназначенных для работы на ПК, входят три основных компонента: командный язык, интерпретирующая система или компилятор для преобразования команд к выполнимому виду и средства взаимодействия пользователя с СУБД (интерфейс пользователя).

Командный язык служит для выполнения и обеспечения требуемых операций с данными - позволяет создавать структуры файлов БД и манипулировать данными, создавать прикладные программы, экранные формы ввода и вывода информации и т.д. Структура и возможности языка в значительной степени определяют облик конкретной СУБД, ее возможности. В состав командного языка ПК также обычно входят специальные команды по установке параметров и состояний системы (SET-команды), а также функции, предназначенные для различных видов обработки данных и выполнения вспомогательных действий.

В СУБД команды можно выполнять по одной, после набора с клавиатуры, или группами команд в автоматическом режиме, предварительно записанных в специальный (программный) файл. Команды языка СУБД записываются в текстовой форме, близкой к обычному языку. Для того чтобы ПК могла выполнить такую команду, ее надо преобразовать в вид исполнимых машинных команд.

Существует два принципиально различных способа такого преобразования. В первом способе используется интерпретирующая система, которая преобразует поочередно команды в исполнимый вид перед их непосредственным выполнением. Во втором способе сначала вся исходная программа преобразуется (компилируется) в программу из исполнимых машинных команд и затем эта программа выполняется.

Первый способ выполнения команд СУБД имеет то преимущество, что исходная программа занимает сравнительно немного места в памяти. Кроме того, этот способ позволяет выполнять команды по одной в режиме ввода с клавиатуры или в режиме, управляемом с помощью меню. Однако выполнение исходной программы посредством интерпретации команд происходит достаточно медленно. Второй способ в отличие от первого позволяет выполнять программу значительно быстрее, но программа, составленная из машинных команд, занимает значительно больше оперативной памяти.

Объем откомпилированных программ по сравнению с их записью на исходном командном языке СУБД может увеличиться в 7-10 раз. Для уменьшения занимаемой оперативной памяти применяются различные способы, например, создаются оверлейные (перекрываемые) структуры программ и используется динамическое распределение элементов программы в памяти. При организации оверлейной структуры в оперативной памяти находится только выполняемая в текущий момент часть программы (модуль), а другие модули хранятся на диске и вызываются в память по мере потребности в их выполнении.

При динамическом распределении памяти используется похожая идея, но реализуется она в более автоматизированном исполнении и применительно не к модулю в целом, а к его элементам - переменным, массивам и т.д. Однако сравнивая скорости выполнения программ двумя способами, следует также иметь в виду, что при использовании интерпретирующей системы в оперативной памяти ПК одновременно находится и программа этой системы, что снижает выигрыш в памяти. В настоящее время наметилась тенденция к отходу от применения интерпретирующих систем в чистом виде. Стремление повысить скорость выполнения команд в режиме интерпретации привело к необходимости специальной предварительной обработки (преобразования) исходной программы.

В результате такого преобразования, называемого псевдокомпиляцией, создается объектный модуль, подготовленный для ускоренного выполнения команд в режиме интерпретации. В последнее время достигнуты большие успехи в разработке псевдокомпиляторов - программ, обеспечивающих этот процесс. В печати по ПК отмечается, что скоростные параметры некоторых СУБД с псевдокомпиляторами (например, Foxbase 2.10) обеспечивают даже более высокую скорость выполнения команд, чем компиляторы (Clipper). Таким образом, видна тенденция к сближению двух основных способов выполнения команд СУБД. Однако различие между ними все же остается и выбор конкретной СУБД определяется целями ее использования.

Системы управления базами данных с компиляторами программ в основном ориентированы на программистов, создающих сложные прикладные системы, а СУБД с интерпретиру-

ющими системами предназначены для широкого круга пользователей, как овладевающих основами программирования, так и активно повышающих свою программистскую квалификацию, вплоть до уровня профессиональных программистов.

Взаимодействие пользователя с СУБД, в которой используется интерпретирующая система, обычно может осуществляться в режиме, управляемом с помощью меню, и режиме, ввода команд с клавиатуры. Первый режим дает возможность пользователям работать с пакетом СУБД, не зная командного языка. Содержание выполняемых команд записывается в позициях меню на естественном языке. Пользователь выбирает нужную позицию меню и нажимает клавишу исполнения команды. Обычно в меню включают достаточно широкий круг команд языка СУБД, позволяющий выполнять многие операции по обработке данных, однако далеко не все. При использовании режима ввода команд с клавиатуры требуется знать их синтаксис и способы применения. Этот режим применяется уже более подготовленными пользователями. Признаком данного режима СУБД служит какой-либо знак (например, точка в левом нижнем углу экрана ПК), который служит приглашением для набора команды. В дальнейшем для простоты будем называть этот режим просто командным.

Развитие интерфейса пользователя с СУБД в настоящее время идет в направлении включения в режим, управляемый меню все большего количества средств, расширяющих возможности неподготовленных пользователей по управлению данными. Например, последние версии таких СУБД, как dBASE, RBASE и др., содержат генераторы прикладных программ, позволяющие пользователям без знания командного языка создавать программы средней степени сложности. Эти средства могут применяться как неподготовленными пользователями, так и профессиональными программистами для уменьшения трудозатрат при разработке прикладных программ.

Структура и содержание компонентов СУБД определяет ее назначение и круг потенциальных пользователей. Пользователей ПК в зависимости от уровня подготовки и диапазона решаемых ими задач можно разделить на три группы:

пользователи, которым для решения своих задач требуется сравнительно небольшой набор функций пакета СУБД (создание файлов данных, ввод и обновление данных, вывод на печать несложных выходных документов и т.д.);

пользователи, которым необходимо знать многие функции пакета СУБД, в том числе и элементы командного языка, но не требуется создавать сложные программы;

программисты, в задачи которых входит разработка сложных прикладных программ по управлению и обработке файлов БД.

Для пользователей первой группы необходимо включать в состав СУБД режим, управляемый с помощью меню. Пользова-

тели второй группы могут осваивать командный режим. Третья группа пользователей ждет от СУБД достаточно полного набора удобных средств для разработки сложных программ.

Последние версии семейства СУБД dBASE - dBASE III+ и IV отвечают требованиям всех перечисленных групп пользователей. В их состав входит весьма развитый командный язык, называемый также dBASE, который фактически стал стандартом командного языка СУБД реляционного типа для ПК; интерпретирующая система (с псевдокомпилятором в dBASE IV); интерфейс пользователя, включающий режим, управляемый с помощью меню, и командный режим. Одним из фактов признания языка dBASE в качестве стандарта можно считать то, что он взят в качестве основы для разработки СУБД, в которых используются компиляторы, например, СУБД Clipper и Quicksilver.

Структура файлов БД семейства dBASE включает текстовые поля длиной до 254 байт, числовые поля (до 20 байт), поля типа даты, логические поля и поля типа Memo. Командный язык СУБД Clipper создан на основе языка dBASE III+ и включает большинство его команд. Не включены в основном полноэкранные команды, например, команда реализации режима управления с помощью меню Assist, команда вызова справочной информации Help и др. Кроме того, в СУБД Clipper ряд команд dBASE III+ выполняется несколько иначе и введены новые команды. Введенные новые команды в языке Clipper можно рассматривать как развитие языка dBASE III+ в некоторых направлениях, например, в направлении средств разработки меню прикладных программ, которые слабо представлены в dBASE III+.

Однако различие систем команд в целом небольшое, что позволяет программы, созданные в СУБД dBASE с учетом небольших отличий в системе команд, компилировать с помощью СУБД Clipper в исполнимые файлы и затем выполнять их, не используя СУБД dBASE.

Структура создаваемых СУБД Clipper файлов БД аналогична структуре файлов БД dBASE. В то же время возможности СУБД Clipper по сравнению с dBASE III+ в плане организации файлов БД имеют некоторые преимущества. В файлах СУБД Clipper может быть до 1024 поля (128 в dBASE III+), можно вводить до 64 000 переменных (256 в dBASE III+), устанавливать связь между 9 файлами БД (между двумя файлами в dBASE III+). Предусмотрена также возможность создания оверлейных структур разрабатываемых программ.

Пакет СУБД Foxbase ориентирован на пользователей второй и третьей групп. В ранних версиях этого пакета отсутствовал режим работы, управляемый меню. Однако СУБД, как и другие программные продукты для ПК, стремительно развиваются. Приблизительно через 1...1,5 года появляются новые версии или значительно обновленные пакеты. При этом происходит взаимное влияние различных концепций, быстро реализуются новые идеи.

Так, в версии Foxbase+ 2.10 (1988 г.) используется интерфейс пользователя, управляемый с помощью меню (Central), хотя и весьма ограниченный по своим возможностям. В то же время в этой версии расширены средства автоматизированной разработки элементов прикладных программ - введен язык создания макетов (язык генерации генераторов программных файлов) для разработки унифицированных прикладных программ типа экранных форм ввода и вывода, отчетов и т.д., что говорит о сохранении пакетом своей специализации, предназначенной в основном для разработки прикладных программ. Командный язык этой СУБД, как и СУБД Clipper, в основном тот же, что и в dBASE III+, а структура файлов БД полностью совпадает с СУБД dBASE. В отличие от СУБД Clipper для выполнения команд в Foxbase используется интерпретирующая система, что привязывает разработанные прикладные программы к пакету. Однако для выполнения программ предусмотрено использование отдельно загружаемого интерпретатора, что уменьшает потребность в оперативной памяти. Перед выполнением программ осуществляется их псевдокомпиляция с целью создания объектного исполняемого модуля. При этом разработчики добились такого совершенства подготовки объектного модуля, что в настоящее время пакет Foxbase является наиболее "быстрым" среди СУБД этого класса.

По своей структуре и реализации версии СУБД R:BASE, выпущенные до 1988 г., были больше ориентированы на пользователей первой и второй групп. Последняя версия этой СУБД может использоваться и профессиональными программистами, так как для ее командного языка разработан компилятор. В качестве командного языка используется язык типа SQL (структурированный язык запросов, специально разработанный для выполнения сложных запросов на поиск и отбор данных из БД небольшим набором команд).

В состав СУБД также входит интерпретирующая система и реализован режим управления файлами БД с помощью меню. В состав одной БД может входить до 80 файлов (таблиц), включающих различные типы полей: числовые (целые, действительные числа, удвоенной точности), поля даты и времени, текстовые поля двух типов: text (длиной до 1500 байт) и note (длиной до 4092 байт). Общее число полей может достигать 800. Режим, управляемый с помощью меню, позволяет создавать и изменять структуры файлов БД, разрабатывать экранные формы ввода и вывода информации, формировать запросы на выборку информации из нескольких файлов БД, генерировать отчеты. В последнюю версию входит также генератор прикладных программ. К положительным сторонам этого пакета относится возможность размещения в одной записи довольно больших объемов текстовой информации, значительное количество полей

в файлах БД, весьма перспективный язык манипулирования данными SQL.

По своим возможностям СУБД Paradox является одним из наиболее развитых пакетов СУБД реляционного типа и предназначен также в основном для пользователей первой и второй группы для решения широкого круга задач, хотя может использоваться и опытными программистами с целью создания сложных программ обработки данных. В состав СУБД Paradox входит командный язык, интерпретирующая система и интерфейс пользователя, управляемый меню. В структуру БД могут включаться текстовые поля длиной до 254 байт, числовые поля, поля типа даты и логические поля фиксированной длины.

Командный язык в основном имеет ту же логику выполнения операций, что и языки семейства dBASE, но представлен своим набором команд. Специфика командного языка Paradox заключается в том, что он интегрирован со своей интерпретирующей системой, которая может находиться в нескольких режимах работы, в результате чего некоторые команды выполняются по-разному в зависимости от режима работы интерпретирующей системы. Поэтому для командного языка СУБД Paradox, пока нет компилятора, что ограничивает область применения этого пакета.

Отличительной особенностью данной СУБД является мощный интерфейс пользователя, управляемый меню. В состав этого интерфейса входят средства создания экранных форм ввода и вывода информации, удобная система формирования запросов на поиск, отбор и вывод данных из нескольких взаимосвязанных файлов БД, генератор отчетов и большой набор вспомогательных средств. Особенно развитыми являются средства формирования запросов на поиск и отбор данных с помощью специальных шаблонов (так называемые запросы по образцу), которые упрощают эту работу для неподготовленных пользователей. Способ формирования запросов по образцу является в настоящее время стандартом интерфейса пользователя в части поиска данных и распространяется в другие СУБД.

Режим работы СУБД Paradox, управляемый меню, позволяет формировать и выполнять довольно сложные последовательности операций (сценарии), включая условия ветвления процесса по созданию файлов БД, отбору требуемых данных и вывод их на экран и принтер. При выполнении сценариев автоматически определяется наиболее оптимальный способ поиска и отбора данных, что можно рассматривать как реализацию некоторых элементов искуственного интеллекта. Несмотря на то, что команды в пакете выполняются интерпретирующей системой, отмечается весьма высокая скорость обработки больших файлов БД. В табл. 1.1 даны сравнительные характеристики СУБД реляционного типа.

Таблица 1.1. **Характеристики некоторых СУБД реляционного типа**

Характеристика	Paradox 2.0	dBASE IV 1.0	dBASE III+	R:base for DOS 2.11	Foxbase 2.10
<u>Ввод и проверка данных</u>					
Командный режим и режим, управляемый меню	Есть	Есть	Есть	Есть	Есть
Количество экранных форм на один файл БД	15	ОПП	ОПП	ОПП	ОПП
Количество строк в одной экранной форме	320	32000	250	115	"
Проверка данных с помощью специальной таблицы	Есть	Есть	В	Есть	В
Проверка диапазона вводимых данных	"	"	Есть	"	Есть
Проверка дублирования индексных выражений	"	"	Нет	"	В
Проверка времени ввода данных	В	"	"	"	"
<u>Средства разработки</u>					
Генератор прикладных программ	Есть	"	Есть	"	Есть
Функции, создаваемые пользователем	"	"	Нет	Нет	"
Макроопределения	"	"	"	Есть	"
Отладчик программ	"	"	"	Нет	Нет
Псевдокомпилятор	Нет	"	"	"	Есть
<u>Поиск и обработка данных</u>					
Связь файлов по типу "одна-несколько" записей	"	"	"	Есть	"
Связь файлов по типу "несколько-одна" запись	Есть	"	"	"	"
Связь файлов по типу "несколько-несколько" записей	Нет	Нет	"	"	"
Обновление нескольких связанных файлов в одной экранной форме	"	В	В	"	"
Связь файла с самим собой	Есть	Есть	Нет	"	Нет
Запросы на языке SQL	Нет	"	"	"	"
Команда восстановления данных	"	"	"	Нет	"
Запросы в форме шаблона (по образцу)	Есть	"	"	Есть	Есть
<u>Генерация отчетов</u>					
Отчеты из нескольких файлов	Нет	"	"	"	"
Создание исходного программного файла	"	"	"	Нет	"
Создание верхних колонтитулов в отчете	Есть	"	"	Есть	"

Окончание табл. 1.1.

Характеристика	Paradox 2.0	dBASE IV 1.0	dBASE III+	R:base for DOS 2.11	Foxbase 2.10
Создание нижних колонтитулов в отчете	"	"	"	"	Нет
Создание итогов в отчете	"	"	"	"	Есть
Создание марок типа письма	"	"	Есть	"	"
<u>Общие данные</u>					
Размер файла БД в записях (триллион)	2	1	1	2	1
Количество полей в записи	255	255	128	800	128
Количество символов в записи	4000	4000	4000	4096	4096
Файлов (таблиц) в одной БД	ОПП	ОПП	ОПП	80	ОПП
Количество индексов на один файл БД	255	47	7	80	7
Количество полей в одном индексе	"	100	100	800	37
Количество связанных файлов	ОПП	8	8	5	10
Максимальное число открытых файлов БД	"	10	10	80	10
Максимальное число переменных	"	15000	256	ОПП	3600
Двухмерные массивы	Нет	Есть	Нет	Нет	Есть
Шифрование данных	Есть	"	Есть	"	Нет

Обозначения: ООП - характеристика, ограниченная только емкостью оперативной памяти; В - характеристика, реализуемая программно.

Национальная лаборатория США по тестированию программных продуктов (NSTL) провела анализ быстродействия работы СУБД Foxbase+ (версия 2.10), dBASE IV (версия 1.0) и dBASE III+ на ПК Compaq 386 с оперативной памятью емкостью 1 Мбайт, жестким диском емкостью 130 Мбайт и тактовой частотой 20 Мггц. Для проведения тестирования использовались БД банковских счетов и имитировалась работа кассиров в отделениях банка розничной торговли по выполнению стандартных операций типа дебет/кредит.

Проведены шесть тестов скорости выполнения следующих операций:

I тест - индексирование четырех файлов БД: файла счетов (50000 записей), файла кассиров (1000 записей), файла отделений банка 100 записей), файла сделок (1000 записей);

II тест - выполнение 1000 стандартных банковских операций с файлами без индексирования;

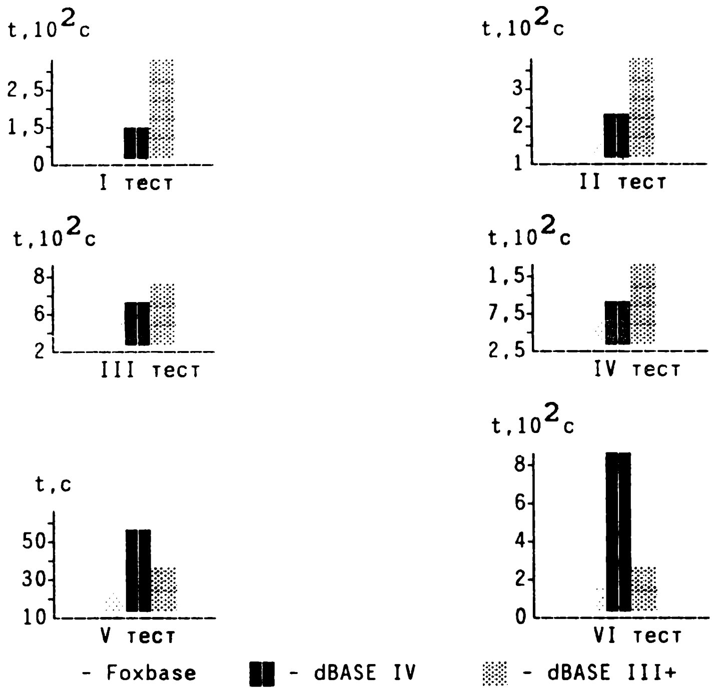


Рис. 1.1. Время выполнения тестов

III тест - выполнение 1000 стандартных банковских операций с индексированным по четырем полям файлом из 1000 записей;

IV тест - выполнение 1000 стандартных банковских операций с индексированным по четырем полям файлом из 10000 записей;

V тест - подготовка отчета с отбором 1000 записей индексированного файла счетов;

VI тест - подготовка отчета, аналогичного предыдущему, с отбором записей без индексирования файла счетов.

На рис. 1.1 приведены данные о времени (в секундах) выполнения этих тестов.

Увеличение времени выполнения теста III по сравнению с тестом II связано с тем, что при добавлении записей с данными о сделках в тесте III затрачивалось время на обновление индексов. При обновлении данных в индексированных файлах БД затрачивается дополнительное время на переиндексирование, но впоследствии применение индексирования дает выигрыш при операциях по поиску и отбору данных, например в случае подготовки отчетов, что видно из тестов V и VI.

1.2. ОБЩАЯ ХАРАКТЕРИСТИКА СУБД dBASE IV

Сравнительный анализ некоторых СУБД реляционного типа показывает, что среди них наиболее универсальными являются СУБД семейства dBASE. Последняя версия этого семейства dBASE IV включает существенно улучшенный командный язык. В нем сохранены все команды языка dBASE III+, многие из которых значительно усилены, а также есть совершенно новые команды и функции, включая команды по организации меню прикладных программ, обработке файлов БД, работе в среде ЛВС и т.д. Как уже упоминалось, в языке dBASE III+ слабо представлены команды для разработки меню прикладных программ. Обычно вместе с dBASE III+ для разработки таких меню используются сопутствующие сервисные пакеты типа Flash-Code, User Interface и др. Расширение языка dBASE IV командами разработки меню прикладных программ устраняет существенный недостаток таких возможностей в СУБД dBASE III+.

В языке dBASE IV также предусмотрены средства ввода и вывода данных в окна на экране, что является значительным развитием системы команд. Особенно усилены в dBASE IV по сравнению с dBASE III+ возможности работы с полями типа Memo. Предусмотрен поиск по этим полям, вывод информации из них в окна, чего не было в dBASE III+.

Кроме этого языка в состав пакета dBASE IV введен дополнительный командный язык SQL, специально предназначенный для выполнения сложных запросов на выборку данных из файлов БД. Эти команды могут выполняться в режиме ввода с клавиатуры или автоматически в виде процедур, вызываемых из программ, написанных на языке dBASE IV. Сочетание развитого языка dBASE IV для создания удобного интерфейса пользователя в прикладных программах и языка запросов SQL позволяет облегчить разработку сложных программных комплексов обработки данных. Процедуры, написанные на языке SQL, могут переноситься в другие СУБД, использующие этот же язык программирования, что расширяет сферу применения пакета СУБД dBASE IV.

В пакете dBASE IV, как и в ранних версиях этого семейства СУБД, используются интерпретирующая система и интерфейс пользователя с режимом, управляемым с помощью меню, и режимом выполнения команд с клавиатуры. В СУБД семейства dBASE, начиная с версии III+, введены средства для работы в среде ЛВС. При работе в этой среде должна быть исключена возможность обращения пользователей, имеющих доступ к общим файлам, к одной и той же информации в одно и то же время. В dBASE IV значительно улучшены средства разрешения конфликтов, возникающих в такой ситуации. Введены средства автоматического захвата файлов и записей при выполнении

команд по обновлению файлов, а также специальные команды, позволяющие осуществить захват файлов и записей в них.

Представление о возможностях пакета dBASE IV можно получить при знакомстве с управляющим центром (УЦ), представляющим собой дальнейшее развитие интерфейса пользователя СУБД семейства dBASE, управляемого с помощью меню. По сравнению с dBASE III+ этот интерфейс настолько переработан, что его можно считать совершенно новым продуктом. В новом интерфейсе систематизированы и расширены функции пакета, существенно улучшено оформление меню на экранах и организация процесса управления БД.

По своим возможностям этот интерфейс примерно равен аналогичным средствам Paradox, а в некоторых отношениях и выше. Например, в состав УЦ помимо обычных компонентов входит генератор прикладных программ, с помощью которого пользователи, не зная команд, могут разрабатывать программы средней степени сложности. Специально для профессиональных программистов для разработки сложных программных комплексов в версии 1.1 пакета dBASE IV фирма Ashton-Tate предполагает поставлять компилятор для создания исполнимых модулей, которые можно выполнять на ПК без пакета dBASE IV только под управлением ОС.

Выше уже были даны некоторые характеристики СУБД семейства dBASE, которые относятся и к пакету dBASE IV. Важной характеристикой является скорость поиска и отбора данных. В реляционных СУБД обычно используются два основных способа поиска данных: последовательный поиск записей и поиск по индексу. При последовательном поиске данных осуществляется перебор всех записей файла БД.

Для поиска по индексу предварительно нужно создать вспомогательный индексный файл. Индексный файл содержит номера записей файла БД, которые упорядочены по возрастанию (или убыванию) индексного выражения, представляющего собой одно поле файла БД или комбинацию нескольких полей. Эти поля объединяются с помощью операторов отношений, разрешенных для соответствующих полей (более подробно об операторах отношений см. гл. 3).

В случае использования индексных файлов поиск заданного значения поля (или комбинации полей) выполняется с помощью номеров записей индексного файла с использованием метода деления отрезка пополам, что значительно ускоряет этот процесс. С одним файлом БД может быть связано несколько индексных файлов, что позволяет осуществлять поиск по различным полям (или их комбинациям).

Файлы БД постоянно хранятся во внешней памяти ПК - обычно на магнитных дисках. Для выполнения действий с файлом БД его записи считываются в оперативную память ПК. Перед считыванием осуществляется подготовка файла, которая называ-

ется открытием файла. Для эффективного использования данных в СУБД часто требуется обрабатывать одновременно несколько файлов. Перед этим их все надо открыть. Максимальное количество одновременно открытых файлов является одним из важнейших показателей возможностей конкретной СУБД. Кроме файлов БД в оперативной памяти ПК в течение сеанса работы с СУБД может размещаться сравнительно небольшой объем данных в виде так называемых переменных. Для них отводятся определенные участки оперативной памяти. В переменной может содержаться любой вид данных, предусмотренных в файлах БД. Переменные могут служить промежуточным хранилищем данных (буфером) при вводе их с клавиатуры в файлы БД или между файлами БД, а также и для других целей, например выполнения арифметических и прочих операций. В dBASE III+ используются одномерные массивы переменных, тогда как в dBASE IV введены двумерные массивы.

Система индексирования полей в dBASE IV построена более удобно, чем в dBASE III+. Индексные файлы в отличие от dBASE III+ объединяются в один множественный индексный файл, связанный с соответствующим файлом БД. Этот файл открывается автоматически при открытии связанного с ним файла БД, а переход на различные индексы (называемые указателями индексов в множественном индексном файле) осуществляется без повторного открытия и закрытия неиспользуемых индексов, что ускоряет процесс обработки данных. Всего может быть создано 47 указателей индексов в одном множественном индексном файле.

По сравнению с dBASE III+ в режиме dBASE IV, управляемом с помощью меню, реализовано значительно больше сервисных операций. В число таких операций входят, например, экспорт и импорт файлов других пакетов (Lotus 1-2-3, Framework II и др.), выполнение команд DOS без выхода из пакета, выход в DOS с возвращением обратно, копирование, удаление, перемещение и переименование файлов, высвечивание директориев дисковых устройств, расцветка областей экрана, создание макроопределений для автоматического выполнения действий, эквивалентных нажатию клавиш на клавиатуре и т.д.

Система управления базами данных dBASE IV значительно усилена по сравнению с dBASE III+ по многим параметрам (ниже в скобках указаны данные для dBASE III+). В частности, можно создавать двухмерные массивы переменных общим объемом до 1170 элементов, может быть открыто до 99 (16) файлов, увеличено максимальное количество переменных - 15000 (256), количество полей в файлах БД - 255 (128), число открытых индексных файлов - 10 (7) и т.д. Как уже отмечалось выше, командный язык dBASE IV значительно расширен и улучшен по сравнению с командным языком dBASE III+.

Включение в язык dBASE IV новых команд позволило существенно улучшить следующие функции пакета СУБД семейства dBASE: разработку иерархических систем меню прикладных задач; разработку экранных форм ввода и вывода данных; организацию ввода и вывода информации в отдельные окна на экране; вывод данных на печать; широкие возможности по поиску и обработке данных в файлах БД; обеспечение работы в среде ЛВС и др.

Особо следует отметить в dBASE IV встроенный отладчик программ, который позволяет перехватывать практически любую ошибку и не дать ей распространиться на данные и программы. Сделанные улучшения позволяют разрабатывать с помощью пакета dBASE IV большие программные комплексы, обеспечивающие сложную обработку информации в сочетании с удобством работы при эксплуатации.

К числу недостатков пакета dBASE IV можно отнести сравнительно медленное выполнение ряда операций интерпретирующей системой (например, генерацию отчетов), а также увеличенную потребность в оперативной и дисковой памяти ПК по сравнению с другими СУБД (для загрузки пакета требуется не менее 640 кбайт оперативной памяти и около 4 Мбайт памяти на диске). Системы управления базами данных семейства dBASE прошли уже весьма длительный путь эволюции. Ввиду безусловно положительного требования совместимости поздних версий с ранними в dBASE IV сохраняются все команды из dBASE III+, нужда в ряде из которых полностью отпадает, так как введены новые, более удобные команды. Это увеличивает общий объем пакета. В целом пакет весьма громоздкий и для изучения всех его возможностей нужно проработать довольно большой объем материала.

Однако недостатки пакета dBASE IV являются продолжением его достоинств. Многофункциональность, большой объем и разнообразие средств управления и обработки БД, высокая степень "дружественности" интерфейса, предназначенного для пользователей любого уровня подготовки, и развитая среда для профессионального программирования позволяют считать dBASE IV в настоящий момент одним из ведущих пакетов универсального предназначения среди СУБД реляционного типа.

Глава 2
РАБОТА С УПРАВЛЯЮЩИМ ЦЕНТРОМ (CONTROL CENTER)

2.1. ОБЩИЕ СВЕДЕНИЯ ОБ УПРАВЛЯЮЩЕМ ЦЕНТРЕ

Представление о возможностях пакета dBASE IV можно получить при знакомстве с управляющим центром (УЦ), реализующим интерфейс пользователя, управляемый меню. В состав УЦ входят шесть меню:

меню **Data** ("Данные") для создания, изменения структуры файлов БД, ввода, обновления и отображения данных на экране;

меню **Queries** ("Запросы") для формирования и выполнения запросов на выборку и обновление данных;

меню **Forms** ("Формы") для разработки экранных форм ввода и вывода информации;

меню **Reports** ("Отчеты") для разработки форм отчетов и вывода их на печать;

меню **Labels** ("Марки") для разработки форм марок и вывода их на печать;

меню **Applications** ("Прикладные программы") - генератор разработки прикладных программ.

Функции УЦ и расположение меню на экране отражают последовательность работы пользователя при разработке прикладной информационной задачи. Работа обычно начинается с создания структуры файлов БД, которые требуются для ее решения. Разработка общей структуры БД представляет собой творческий процесс, в котором необходимы как знания прикладной информационной задачи, так возможностей конкретной СУБД. При этом определяется структура файлов БД, их взаимосвязь, способы обработки информации и формы вывода данных и т.д.

После создания структур файлов БД прикладной задачи требуется ввести в них данные. Ввод может быть осуществлен с клавиатуры ПК или из другого файла БД. В процессе эксплуатации информационной задачи возникает необходимость обновления и ввода новых записей файлов БД. Эти операции могут быть выполнены с помощью меню УЦ несколькими способами. При обновлении данных вручную с клавиатуры можно использовать меню "Данные". В случае обновления больших массивов данных из других файлов целесообразно формировать специальные запросы на обновление данных с помощью меню "Запросы".

Ввод и обновление данных в меню "Данные" выполняется на экране в стандартных формах. Эти формы не всегда вполне удобны для пользователя. В УЦ предусмотрено меню "Формы", позволяющее разрабатывать собственные экранные формы для ввода и вывода информации, в которых можно удобно расположить поля записей файла БД, ввести пояснения и другую вспомогательную информацию, организовать контроль ввода данных и т.д.

После того как разработана структура файлов данных прикладной задачи, отработаны экранные формы для ввода и обновления данных, для формирования выходных документов требуется отобрать необходимую информацию. Отбор информации из одного или нескольких файлов БД можно выполнить с помощью меню "Запросы". В состав этой информации могут входить не только данные, непосредственно находящиеся в файлах БД, но и данные, получаемые в результате различных способов обработки записей, например, итоговые данные по группам значений полей файлов БД, средние, максимальные и минимальные значения групп, количество записей в группах, а также результаты различных других арифметических операций с значениями числовых полей файлов БД. Отобранные данные можно выводить на экран в разработанных ранее формах или на печать.

Меню "Запросы" позволяет также связывать между собой несколько файлов БД, что дает возможность при разработке структуры файлов БД прикладной задачи избежать дублирования информации и тем самым сэкономить внешнюю память, увеличить быстродействие задачи и ускорить процесс обновления данных.

Завершающий этап процесса обработки информации - вывод полученных данных на экран, принтер или внешние носители. Важной стороной данного этапа является представление выходных данных в форме, удобной для анализа и оценки. Для генерации форм выходных документов предназначено меню "Отчеты". Это меню позволяет создавать различные формы выводимых документов как по содержанию, так и по размещению материала с целью его наглядного восприятия.

Можно выводить на печать уже обработанные данные, не изменяя при этом саму БД (примерно так, как это делается в запросах на отбор информации). В частности, можно получать итоговые численные значения в группах однородной информации, выполнять статистическую обработку и арифметические действия над полями данных. Кроме того, предусмотрена возможность вывода на печать данных, сохраненных на диске после отбора с помощью созданных ранее запросов. Для создания сложных выходных документов можно использовать данные, полученные в результате выполнения запросов, которые затем подвергаются

дальнейшей обработке уже в ходе подготовки вывода документа на печать.

Таким образом, меню УЦ позволяет выполнять все этапы процесса обработки информации, начиная от ее записи на носители и завершая выдачей необходимых выходных документов. Этот процесс реализуется посредством выполнения пользователем действий с помощью меню УЦ. Число этих действий может быть достаточно велико. Для автоматизации обработки данных обычно требуется разработать программу на командном языке СУБД, которая затем выполняется в пакетном режиме. Чтобы составить такую программу, надо знать систему команд СУБД. Меню "Прикладные программы" позволяет создать программу работы СУБД, не зная системы команд. В состав этой программы могут входить разнообразные меню пользователя для прикладной задачи, которые обеспечивают удобство и наглядность ее эксплуатации и реализацию всех требуемых операций в автоматическом режиме.

Чтобы различать файлы, используемые в СУБД, им присваиваются имена, а для определения типа файла к имени присоединяется через точку его расширение. Например, все файлы БД в семействе СУБД dBASE имеют расширение **.dbf**, индексные файлы dBASE III+ - расширение **.ndx**, а множественные индексные файлы в dBASE IV - **.mdx**, программные файлы - **.prg** и т.д.

Разработанные с помощью УЦ экранные формы, запросы, отчеты, марки и прикладные программы представляют собой программные файлы, которые записываются в виде файлов с различным расширением, характеризующим его тип. Эти файлы, как и файлы БД и связанные с ними индексные файлы, хранятся в каталогах. Каталог - это файл, в котором хранятся имена и место физического расположения группы файлов, объединенных определенной целью использования, например, для одной прикладной задачи, для одного пользователя и т.д. Каталог сходен с директорием, но в отличие от последнего не образует иерархической структуры. В директорий включаются файлы по месту их расположения на диске, а в каталог - по назначению. В каталог могут быть включены файлы из различных директориев и дисков. Для работы с каталогами в УЦ имеются удобные меню, которые описаны в разд. 2.3.

Здесь не описывается процедура установки пакета, которую можно найти в техническом описании. После установки вызов пакета может быть выполнен различными способами, в том числе с использованием систем меню сервисных пакетов типа Norton Commander и др. В любом случае вход в пакет осуществляется запуском основного исполняемого модуля **dbase.exe** из среды операционной системы. После входа в dBASE IV по умолчанию устанавливается режим ввода команд с клавиатуры (высвечивается точка в левом нижнем углу - приглашение для ввода

команды). Переход из этого режима в режим, управляемый меню, осуществляется командой **Assist** или нажатием функциональной клавиши **F2**. На рис. 2.1 показан экран УЦ.

На экране УЦ в верхней строке высвечивается горизонтальное меню, предназначенное для выполнения различных операций с файлами. После входа в УЦ всегда высвечивается меню сервисных функций, показанное на рис. 2.1. Это меню служит для выполнения операций общего назначения. В следующей строке помещается английское название управляющего центра, затем имя текущего каталога, а далее панель, в которой высвечиваются имена файлов из текущего каталога. Под панелью высвечивается имя файла, на котором установлен курсор в текущий момент и словесное описание этого файла. В нижней части экрана размещается информационная строка, в которой дается вспомогательная информация (наименования клавиш, выполняющих основные операции) и строка сообщений, появляющихся при выполнении позиций меню (эта строка на рис. 2.1 не показана).

Панель УЦ состоит из шести окон. Окно **Data** содержит имена файлов БД. Окно **Queries** показывает имена файлов запросов на выборку и обновление данных. В окне **Forms** высвечиваются имена файлов экранных форм ввода и вывода информации. Окно **Reports** содержит имена файлов форм отчетов, а окно **Labels** - имена файлов форм марок.

Крайнее справа окно **Applications** высвечивает имена программ, разработанных с помощью генератора прикладных программ или любых текстовых файлов. После входа в УЦ курсор устанавливается в одно из окон и высвечивается меню сервисных функций, которые описаны в разд. 2.3. При этом меню находится в неактивном состоянии, т.е. высвечиваются только горизонтальные позиции в верхней части экрана. Активизация этого меню (и любых других меню) осуществляется клавишей **F10**.

Признаком активизации служит появление вертикального меню, прикрепленного к одной из позиций горизонтального меню. Деактивизация, сопровождающаяся удалением прикрепленного вертикального меню, осуществляется клавишей **Esc** (вообще клавиша **Esc** служит также для отмены любого начатого, но не оконченного действия и возвращения к предыдущему состоянию). Передвижение по окнам панели УЦ осуществляется клавишами со стрелками **"Вправо"** и **"Влево"**, а внутри окна клавишами **"Вверх"** и **"Вниз"**.

Для создания файлов соответствующих типов в окнах панели УЦ имеется позиция **<create>** (создать). После выбора этой позиции высвечиваются экраны, предназначенные для выполнения операций по созданию файла соответствующего типа. Эти же экраны раскрываются и при выборе имени файла в окне.

```
Catalog  Tools   Exit                             12:17:08 pm

                     dBASE IV CONTROL CENTER
                 CATALOG: D:\DBASEIV\PRIMER\PRIMER.CAT

   Data       Queries     Forms      Reports     Labels  Applications
 <create>    <create>    <create>   <create>    <create>   <create>
 --------    --------    --------   --------    --------   --------
   BD1        Q01         F1          R1          L1         AVTO
   BD2        Q02         F2                                 TEXT1
   BD3        Q03         F3                                 TEXT2
   BD4        Q04         F4
              Q05
              Q06
              Q07
              Q08
              Q09

 File: BD1.DBF
 Description: Данные о водителях

Help:F1 Use:<┘ Data:F2 Design: Shift-F2 Quick Report:Shift-F9 Menus:F10
```

Рис. 2.1. Экран управляющего центра

В верхней части экрана создания (модификации) файла высвечивается основное (горизонтальное) меню того окна, из которого был выбран файл. Меню каждого окна имеет иерархическую структуру. Оно состоит из основного горизонтального меню и системы вертикальных меню, прикрепленных к позициям меню вышестоящего уровня. В любом меню выполняемыми обычно являются позиции самого нижнего уровня. Неисполняемые позиции меню, которые служат для перехода на нижеследующий уровень, обычно помечаются знаком треугольника слева от названия позиции. Выход на выполняемую позицию осуществляется посредством последовательного раскрытия меню нижестоящих уровней. Ниже в тексте под термином меню будет подразумеваться как меню определенного окна в целом, так и неисполняемые позиции меню, к которым прикреплены вертикальные меню нижеследующего уровня (обычно позиции основного - горизонтального меню окна). Из контекста будет ясно, о каком меню идет речь.

Для того чтобы быстрее научиться работать с УЦ, нужно прежде всего освоить операцию выбора, которая является главной среди остальных действий, выполняемых на клавиатуре. Операция выбора состоит из двух основных действий: передвижения курсора на требуемую позицию и нажатия клавиши **Enter** (выполнение действия, связанного с операцией выбора). Передвижение курсора на одном уровне меню осуществляется навигационными клавишами (**Вправо**, **Влево**, **Вверх**, **Вниз**, **PgUp**,

PgDn, **Home** и **End**). Раскрытие меню нижеследующего уровня осуществляется клавишей **Enter**, естественно, после установки курсора на соответствующую позицию. Чаще всего операция выбора применяется при выборе позиции меню и файла в окне **Data**. Кроме того, при работе с УЦ нередко используется операция выбора необходимого значения (оператора, переменной, данных, имен файлов и т.д.) из списка значений. Список значений вызывается на экран клавишами **Shift-F1** или при выборе позиции меню. При этом может высвечиваться сразу весь список или же одно значение из этого списка. Например, при вызове клавишами **Shift-F1** высвечивается сразу весь список. Если высвечивается только одно значение, то другие можно вызвать последовательным нажатием клавиши **Пробел**. После установки курсора на требуемое значение или вызова клавишей **Пробел**, его выбор также завершается клавишей **Enter**. Таким образом, операция выбора из списка также состоит из двух основных действий, но может включать дополнительные действия по нажатию клавиши **Пробел**.

Отметим, что обычно выбору значения из высвечиваемого списка есть альтернатива - можно просто ввести требуемое значение с клавиатуры. Однако нужно подчеркнуть, что система предложения на выбор значений в dBASE IV специально предусмотрена для избежания ошибок и сокращения времени ввода. Поэтому целесообразно выбирать требуемые значения из предоставляемого списка, а не вводить с клавиатуры.

Иллюстрацию возможностей УЦ лучше всего проводить на конкретной задаче. Ниже будет рассмотрена информационная задача, на примере которой будут продемонстрированы функции УЦ.

2.2. ЗАДАЧА УЧЕТА И КОНТРОЛЯ РАБОТЫ ТРАНСПОРТНЫХ СРЕДСТВ

Рассмотрим задачу учета перевозок грузов автотранспортным предприятием, в парке которого имеются автомобили различных марок. Требуется автоматизировать учет и контроль работы транспортных средств по следующим комплексам данных: маршрутам и времени перевозок, перевозимому грузу, состоянию и местонахождению автомобиля и его характеристикам, персональным данным о водителях.

Маршрут и время перевозок включают: пункт погрузки; пункт разгрузки; пройденное расстояние; дату выезда; дату прибытия в пункт разгрузки.

По перевозимому грузу учитываются следующие данные: наименование груза; организация, отправляющая груз; организация, получающая груз; вес груза.

Состояние автомобиля определяется его исправностью или неисправностью, а местонахождение зависит от того, где он находится - в рейсе, в автопарке или в капитальном ремонте. Технические характеристики автомобиля включают: среднюю скорость движения, км/ч; грузоподъемность, т; расход топлива л/км; марку автомобиля; номерной знак.

Данные о водителе содержат: фамилию, имя, отчество; анкетные данные (автобиографию); номерной знак автомобиля, на котором работает.

На основании перечисленных выше данных требуется периодически выдавать статистику количества перевозимого груза, пройденного километража, расхода топлива, состояния автомобилей и т.д. Если попытаться представить всю информацию в виде одного файла БД, то такой файл получится очень громоздким, многие данные будут дублироваться, будет очень сложно выполнять операции по поиску, отбору и обновлению данных, в силу чего работать с таким файлом практически невозможно. Поэтому для обеспечения эффективного выполнения операций по обработке информации необходимо сначала провести анализ и определить структуру БД в виде связанных между собой файлов БД. Анализ нашей информационной задачи показывает, что для учета и контроля работы транспортных средств целесообразно создать четыре файла БД с именами: **BD1**, **BD2**, **BD3** и **BD4**:

BD1 - данные о водителях;
BD2 - имеющиеся на предприятии автомобили;
BD3 - перевозимый груз и маршруты движения;
BD4 - технические характеристики автомобилей.

Файл БД **BD1** содержит поля с именами:

NOM - табельный номер водителя;
FAM - фамилия;
NAM - имя;
OTC - отчество;
D_R - дата рождения;
S_R - стаж работы;
KAT - категория;
AUTO - автобиография;
N_Z - номерной знак автомобиля, на котором водитель работает.

Файл БД **BD2** включает поля:

N_Z - номерной знак автомобиля;
SOS - техническое состояние;
MAR - марка автомобиля;
G_N - местонахождение (в рейсе, в автопарке, в капитальном ремонте).

Файл БД **BD3** содержит поля:

N_G - наименование груза;
O_P - откуда перевозится груз;
D_V - дата выезда;

K_D - куда необходимо доставить груз;
D_P - дата прибытия;
RAS - пройденное расстояние;
VRM - время, затраченное на дорогу;
VES - масса груза;
RGR - фактический расход горючего;
N_Z - номерной знак автомобиля, перевозящего груз.

Файл БД **BD4** включает поля:
MAR - марка автомобиля;
SKR - средняя скорость движения, км/ч;
GRZ - грузоподъемность, т;
RSX - номинальный расход топлива, л/км.

Общие поля файлов БД служат для их связки. На основе этих связей образуется интегрированная база данных, в которой хранится информация о водителях, транспортных средствах и перевозимых грузах. Схема связи файлов БД представлена на рис. 2.2.

В качестве самостоятельной работы попытайтесь расширить эту задачу в отношении учета и хранения дополнительных данных.

Например, файл БД **BD1** можно дополнить полями, учитывающими информацию для отдела кадров и вводить в него данные не только на водителей, но и на остальных служащих предприятия (поля должности, номера отдела и т.д.).

Для финансового отдела можно создать новый файл БД **BD5**, в котором хранится информация о количестве рабочих дней, размере начислений и удержаний, необходимых для расчета заработной платы.

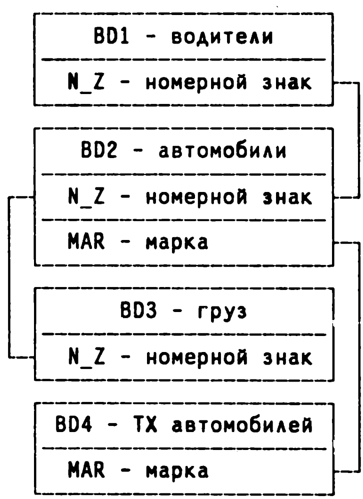


Рис. 2.2. Схема связи файлов БД автопредприятия

Автоматизируя складской учет, можно создать файл БД **BD6** для хранения данных о материально-техническом обеспечении (МТО).

Тогда можно построить базу данных для автоматизированной системы управления (АСУ) автопредприятия, включающую в себя подсистемы учета перевозок, учета автотранспортных средств, материально-технического обеспечения, а также кадровую и финансовую подсистемы. Структурно АСУ автопредприятия можно представить так, как показано на рис. 2.3.

Изучение материалов по dBASE IV базируется на самостоятельной работе по решению примеров, приводимых в последующих разделах. Если читатель встретит трудности, то он может обратиться к учебной дискете, на которой записаны все решения этих примеров.

Решения записаны на учебной дискете в директорие **PRIMER**, который включает четыре файла БД (**BD1**, **BD2**, **BD3**, **BD4**), 18 запросов (**Q01** - **Q18**), четыре экранные формы (**F1** - **F4**), отчет **R1**, марку **L1**, прикладную программу с именем **AVTO** и два текстовых файла **TEXT1** и **TEXT2**. Файлы учебного примера с дискеты целесообразно скопировать на жесткий диск. Предлагаемые примеры можно решать и записывать их результаты в тот же директорий, в котором находятся учебные примеры, считанные с дискеты.

Имена создаваемых Вами файлов должны отличаться от файлов учебной дискеты. При одинаковых именах новый создаваемый файл записывается на место старого, так что файл учебной дискеты будет заменен созданным Вами файлом.

Копирование учебной дискеты на жесткий диск можно выполнить средствами dBASE IV или DOS. Поскольку Вы еще практически не приступили к изучению пакета, покажем, как копировать, находясь в среде DOS. В следующем разделе будет описано как это сделать средствами dBASE IV. Перед копированием целесообразно создать в директории **DBASE** поддиректорий, например с именем **PRIMER**, куда и будет копироваться содержимое директория **PRIMER** дискеты.

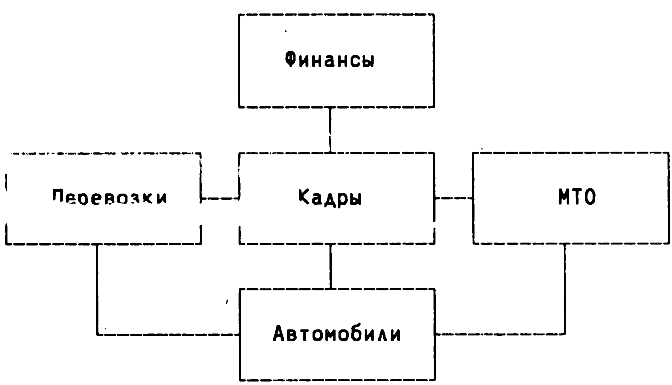


Рис. 2.3. Подсистемы АСУ автопредприятия

Эти операции удобно выполнять с помощью пакета программ Norton Commander (запуск этого пакета, если он не осуществлен при загрузке Вашего компьютера, реализуется в среде DOS вводом с клавиатуры имени **NC** (или **NCSMALL**) и последующим нажатием клавиши **Enter**). Затем, находясь уже в среде пакета Norton Commander, нажмите одновременно клавиши **Alt-F1**, и в высветившемся окне выберите имя логического диска (**C,D,E,F** и т.д.), в котором находится dBASE IV (предположим, что dBASE IV расположен на диске **D:**), и нажмите клавишу **Enter**. В левой панели высветится содержимое выбранного диска. Если курсор находится в правой панели, то с помощью клавиши **TAB** переместите его в левую панель экрана.

Затем установите курсор на директорий **DBASEIV** (на различных компьютерах имя этого директория может варьироваться, например, DBASE4, DB4, DBASE и т.д., что определяется тем, как был задан директорий при установке пакета dBASE IV) и нажмите клавишу **Enter**. Тем самым Вы войдете в директорий **DBASEIV**. Затем нажмите клавишу **F7** для создания поддиректория **PRIMER**. После этого появится окно, в котором записано имя текущего диска и директория, например, **D:\DBASEIV**. Дополните маршрут именем поддиректория с ведущей наклонной чертой влево **D:\DBASEIV\PRIMER** и нажмите клавишу **Enter** (для новой версии Norton Commander текущий директорий в окне не высвечивается, поэтому нужно вводить с клавиатуры весь маршрут **D:\DBASEIV\PRIMER**). Поддиректорий **PRIMER** директория **DBASEIV** создан. Установите курсор на созданный поддиректорий **PRIMER** и нажмите клавишу **Enter** для его раскрытия.

Теперь скопируйте в этот поддиректорий содержимое директория **PRIMER** дискеты. Для этого нажмите клавиши **Alt-F2**, в высветившемся окне выберите диск **A:** и нажмите клавишу **Enter**. В правой панели Norton Commander высветится содержимое дискеты. Клавишей **TAB** перейдите на правую панель с содержимым дискеты, установите курсор на директорий **PRIMER** и раскройте его содержимое клавишей **Enter**. Для выбора всех файлов директория **PRIMER** нажмите клавишу **" + "** (**Gray Plus**) справа от цифровой клавиатуры и затем **Enter**. При этом все выбранные файлы будут выделены желтым цветом. Для копирования файлов с дискеты на жесткий диск нажмите клавишу **F5**, после чего высветится окно, в котором записан маршрут по директориям **D:\DBASEIV\PRIMER**, куда копируется дискета и, убедившись в правильности этого маршрута, нажмите клавишу **Enter**. После этого файлы будут последовательно копироваться с визуальным отображением объема копируемого файла.

Отметим, что при разработке и эксплуатации информационных задач часто требуется выполнять операцию обратного копирования: с жесткого диска на дискету. Эта операция выполняется подобно описанной, с той лишь разницей, что подсвечивать надо уже файлы на жестком диске (которые копируются). Вообще для всех операций копирования действует правило: надо высветить копируемые файлы, безразлично в какой половине экрана и установить клавишей **TAB** курсор на эту половину. После нажатия клавиши **F5** подсвеченные

файлы копируются на диск в директорий, который установлен в другой половине экрана. Отметим также, что подсветка отдельных файлов выполняется клавишей **Ins**, а гашение подсветки отдельного файла - повторным нажатием этой клавиши. Гашение всех подсвеченных файлов осуществляется нажатием клавиши "-", расположенной выше клавиши "+", с помощью которой выполнялась подсветка, и последующим нажатием клавиши **Enter**. Заметим, что при копировании одного файла его имя можно не подсвечивать - достаточно установить на нем курсор.

2.3. СЕРВИСНЫЕ ФУНКЦИИ УПРАВЛЯЮЩЕГО ЦЕНТРА

Перед тем как начать изучать основные функции УЦ, рассмотрим его сервисные функции. При входе в УЦ всегда сначала высвечивается меню сервисных (обслуживающих) функций, включающее три позиции (меню): **Catalog**, **Tools** и **Exit**.

Меню **Catalog** позволяет создавать каталоги файлов. Каталог представляет собой файл, который содержит список файлов (обычно одной предметной области), размещаемых на одном или нескольких дисках, в том числе и на сменных. В каталоге записываются имена файлов, их местоположение и краткое словесное описание. Каталог имеет сходство с директорием, но между ними есть и существенные различия. Директорий так же, как и каталог, представляет собой файл, содержащий список файлов. Но в директории в отличие от каталога файлы размещены на одном диске. Если файл удаляется из директория, то он автоматически удаляется с диска. Удаление же файла из каталога не вызывает его удаления с диска. Один и тот же файл можно занести в несколько различных каталогов.

Важнейшая особенность системы каталогов dBASE IV состоит в том, что после загрузки каталога все его файлы автоматически высвечиваются на панели УЦ. Другой особенностью этой системы является автоматическая каталогизация, т.е. автоматическое помещение в текущий каталог всех создаваемых в текущем сеансе файлов. Третья особенность - это автоматическая фильтрация файлов и размещение их на панели УЦ в соответствующее окно. Так, файлы БД всегда высвечиваются только в окне **Data**, файлы запросов - в окне **Queries** и т.д. Кроме того, если требуется загрузить файлы определенного типа из другого директория в текущий каталог, то в вызванном списке файлов этого директория высвечиваются только файлы этого типа.

При входе в УЦ автоматически загружается каталог, который использовался в предыдущем сеансе. Если раньше не было создано ни одного каталога, то при наличии в текущем директории хотя бы одного файла БД (с расширением **.dbf**) автомати-

чески создается и становится текущим первоначальный каталог с именем **Untitled.cat**. Таким образом, при работе с УЦ практически всегда имеется текущий каталог, в который автоматически записываются все создаваемые файлы.

Если в текущем директории нет файлов БД, то каталог остается пустым до тех пор, пока не будут созданы или скопированы файлы этого типа. Для установки в ходе сеанса другого каталога нужно выбрать позицию **Use a different catalog** меню **Catalog**. При этом высвечивается список имен всех имеющихся в текущем директории каталогов. Выбор требуемого каталога приводит к его загрузке, и он становится текущим. Это отмечается на экране УЦ в верхней его части: после слова **CATALOG**: помещается имя текущего директория (маршрут по директориям) и имя текущего каталога.

Усвоение данного руководства будет глубже, если самостоятельно выполнить все приводимые ниже примеры. Для этого целесообразно создать собственный каталог и открывать его в каждом сеансе учебного курса. В этот каталог будут заноситься все создаваемые Вами файлы.

Перед созданием каталога активизируйте меню сервисных функций нажатием клавиши **F10**. Активизация сопровождается высвечиванием прикрепленного меню. Выберите позицию **Use a different catalog** меню **Catalog**, т.е. установите на нее курсор и нажмите клавишу **Enter**. В правой верхней части экрана высветится список уже имеющихся в текущем директории каталогов. В верхней части списка находится позиция **<create>**, которая служит для создания нового каталога. После выбора этой позиции высвечивается предложение **Enter name for new catalog** (введите имя нового каталога) и окно для ввода имени каталога, в котором уже указан маршрут по директориям, т.е. имя диска и последовательность имен директориев, разделенных косой чертой. Остается только ввести имя каталога, например, свою фамилию или ее аббревиатуру и нажать клавишу **Enter**. После этого высветится экран УЦ, где в качестве текущего будет созданный Вами каталог.

Если имя каталога оказалось неудачным, его можно изменить, выбрав позицию **Modify catalog name**. Стирание имени осуществляется клавишами **Backspace** (удаление символов слева от курсора) или **Del** (удаление символа на позиции курсора). Для удобства поиска своего каталога, например, в случае, если Вы забыли его имя, можно ввести словесное описание каталога, которое будет высвечиваться в левой части экрана при выборе имени каталога из списка. После выбора позиции **Edit description of catalog** меню **Catalog** и нажатия клавиши **Enter** высвечивается окно, в которое с клавиатуры введите описание каталога, например: "Каталог учебных файлов Иванова И.И." В том случае, если каталог оказался не нужным, его можно удалить тем же

способом, каким удаляется любой файл, учтя что файл каталога имеет расширение **.cat** (об удалении файла см. ниже).

Как уже указывалось, в текущий каталог будут заноситься все создаваемые Вами файлы. Однако нередко требуется поместить в каталог другие, ранее созданные файлы. Для этого используется позиция **Add file to catalog**. После выбора этой позиции в правой части экрана высвечивается список файлов и поддиректориев текущего директория. Раскрытие поддиректория выполняется так же, как и меню, - клавишей **Enter**. Для переноса файла из директория в свой каталог нужно его выбрать, т.е. установить курсор на имя файла и нажать клавишу **Enter**. После этого на экране появится окно с предложением ввести словесное описание этого файла, которое будет впоследствии высвечиваться при выборе файла в окне **Data**. Ввод описания файла осуществляется так же, как и ввод описания каталога: вводится с клавиатуры текст, а затем нажимается клавиша **Enter**, что означает завершение ввода текста и его запись на диск. Такое завершение является стандартным для многих операций по вводу данных.

После переноса файла из директория в Ваш каталог его имя появляется в соответствующем окне экрана УЦ, т.е. файл БД появляется в окне **Data**, файл экранной формы - в окне **Forms** и т.д. Если описание файла оказалось неудачным, его можно изменить. Для этого надо установить курсор на имя файла в соответствующем окне и затем выбрать позицию **Change description of highlighted file**. После этого высветится окно для ввода нового или коррекции имеющегося описания файла, которые выполняются так же, как и ввод первоначального описания.

Ненужный файл можно исключить из каталога. Напомним, что это не означает удаления файла с диска. Просто имя удаленного из каталога файла уже не будет высвечиваться на панели УЦ при загрузке данного каталога. Перед исключением какого-либо файла из каталога нужно его сначала закрыть, если он еще не закрыт. Признаком того, что файл закрыт или открыт служит его размещение над чертой в окне на панели УЦ (файл открыт) или под чертой (файл закрыт). Процесс закрытия файла начинается с его выбора, т.е. установки курсора на его имя и нажатия клавиши **Enter**. Затем высвечивается горизонтальное меню, в котором нужно выбрать позицию (крайнюю левую) **Close file**, после чего файл закрывается, что отмечается его перемещением ниже черты на панели УЦ.

При работе с любыми файлами, в том числе с файлами учебных примеров, рекомендуется закрывать файлы, которые в данный момент не нужны. В этом случае работа СУБД будет более стабильной. Практика показывает, что если оставляется открытым ненужный файл, то это может привести к весьма

тяжелым последствиям, вплоть до того, что файл перестанет читаться.

Как читатель уже догадывается, перед выполнением любых манипуляций с файлом нужно установить курсор на его имя в соответствующем окне. Так, для удаления закрытого файла из каталога нужно предварительно установить курсор на его имя и затем выбрать позицию **Remove highlighted file from catalog**. После этого высвечивается предупреждающее сообщение. В этот момент можно еще отказаться от удаления файла из каталога (ответ **No**) или подтвердить удаление (ответ **Yes**). После удаления файла из каталога высвечивается предложение на удаление файла из директория на диске. Это уже означает уничтожение файла. Здесь надо быть осмотрительным и не спешить, так как уничтоженный файл восстановить средствами dBASE IV уже невозможно.

При разработке запросов, форм, отчетов, марок и прикладных программ генерируется несколько типов программных файлов: файл разработки, файл сгенерированных кодов и компилированный файл. Файл разработки служит для последующей генерации второго типа файла - программного модуля, который в свою очередь преобразуется в исполнимый модуль (компилированный) файл. Исполнимый программный файл является конечным продуктом генерации соответствующего объекта (экранной формы, запроса и т.д.). При этом другие файлы (разработки и сгенерированных кодов) при выполнении компилированного файла уже не требуются, однако они будут нужны, если потребуется изменить этот файл. В зависимости от этого определяется необходимость хранения файла разработки и сгенерированных кодов. В табл. 2.1 представлены типы файлов и их расширения.

Таблица 2.1. **Типы файлов**

Файл	Окно УЦ	Файл разработки	Файл кодов	Исполнимый файл
БД	Data	.dbf	-	-
Запрос на выборку	Queries	.qbe	-	.qbo
Запрос на обновление	"	.upd	-	.upo
Выборка dBASE III+	"	.vue	-	-
Экранная форма	Forms	.scr	.fmt	.fmo
Отчет	Reports	.frm	.frg	.fro
Марка	Labels	.lbl	.lbg	.lbo
Прикладная программа	Applications	-	.prg	.dbo
SQL программа	"	-	.prs	.dbo
Программа, созданная генератором прикладных программ	"	.app	.prg	.dbo

В ряде случаев возникает необходимость перевода созданных файлов из одного каталога в другой. Для форм, отчетов и марок можно вводить в каталог только файлы разработки и исполнимые файлы. Файлы сгенерированных кодов создаются и вводятся в каталог автоматически при создании соответствующих объектов (форм, запросов и т.д.). Если в каталог вводится только файл разработки, то файл кодов и исполнимый файл включаются в каталог автоматически.

Можно вводить в каталог только исполнимые файлы, например в случае отсутствия файлов разработки и сгенерированных кодов, однако, как уже указывалось, модифицировать такие исполнимые файлы уже нельзя. В табл. 2.2 показаны файлы, которые можно ввести в каталог и которые высвечиваются в нем.

Меню **Tools** обеспечивает средства для облегчения и удобства работы пользователя на клавиатуре (создание макроопределений), позволяет выходить в DOS из dBASE IV, выполнять команды DOS и возвращаться в dBASE IV, выполнять установки параметров dBASE IV, создавать защиту данных от несанкционированного доступа и схемы ограничения доступа к файлам, экспортировать свои и импортировать внешние файлы, копировать и переименовывать файлы, а также выполнять другие функции.

После выбора меню **Tools** высвечивается вертикальное меню с позициями: **Macros**, **Import**, **Export**, **DOS utilities**, **Protect data**, **Settings**.

Позиция **Macros** служит для создания макроопределений - программ, имитирующих нажатие любой последовательности клавиш клавиатуры, включая вызов меню, выбор его позиций и т.д.

Таблица 2.2. **Типы файлов в каталоге**

Файл	Окно УЦ	Вводимые файлы	Высвечиваемые файлы
БД	Data	.dbf	.dbf
Запрос на выборку	Queries	.qbe,.qbo	.qbe
Запрос на обновление	"	.upd,.upo	.upd
Выборка dBASE III+	"	.vue	.vue
Экранная форма	Forms	.scr,.fmo	.scr,.fmt
Отчет	Reports	.frm,.fro	.frm,.frg
Марка	Labels	.lbl,.lbo	.lbl,.lbg
Прикладная программа	Applications	.prg,.dbo	-
SQL программа	"	.prs,.dbo	-
Программа генератора прикладных программ	"	.app,.prg .dbo	.app
Внешние программы	"	.exe,.com	-

Макроопределение вызывается одной из функциональных клавиш **F1-F9**, комбинацией клавиши **F10** и латинской буквы или несколькимии латинскими буквами (только для системных макроопределений). Для вызова макроопределения с клавиатуры нужно нажать клавишу **Alt** и, удерживая ее, нажать функциональную клавишу или букву.

Затем клавишу **Alt** нужно отпустить и нажать последующие буквы. Макроопределения хранятся в библиотеках. Каждая библиотека записывается на диск с уникальным именем и может включать до 35 макроопределений.

В макроопределения можно включать другие макроопределения и создавать вложенные структуры макроопределений (до 16 уровней). Нельзя делать вложения самих в себя. Для всех меню в dBASE IV имеются системные макроопределения, которые вызываются последовательностью первых букв имен позиций, ведущих к требуемой позиции.

Например, выход из dBASE IV с панели УЦ можно выполнить клавишами **Alt-eq** (клавиша **Alt** нажимается вместе с буквой **e**, затем отпускается). Пользователи могут создавать и модифицировать только макроопределения, вызываемые функциональными клавишами **F1-F9** или комбинацией клавиши **F10** и буквы.

После выбора позиции **Macros** высвечивается меню, позволяющее выполнять операции по созданию макроопределения, его модификации, удалению, копированию, изменению имени и текста макроопределения, вводу в текст макроопределения точек прерывания, сохранению макроопределения в библиотеке на диске и др.

Для создания макроопределения выбирается позиция **Begin recording**. После этого высвечивается таблица с функциональными клавишами **F1-F10** и буквами, которые служат для вызова макроопределений. При первом назначении макроопределения функциональной клавише или комбинации клавиши **F10** и латинской буквы ему присваивается имя - соответственно **F1-F9** или латинская буква. Это имя появляется в соответствующей графе таблицы (в верхней части таблицы имена макроопределений, присвоенные функциональным клавишам, а в нижней части - комбинациям клавиши **F10** и буквы от **a** до **z**). В последующем первоначально назначенное имя можно изменить с помощью позиции **Name** на имя, состоящее из любых вводимых с клавиатуры символов. Следует отметить, что созданные пользователем макроопределения вызываются с клавиатуры только первоначально назначенными функциональными клавишами или комбинацией функциональной клавиши **F10** и буквы, а имя в таблице макроопределений используется только для вызова макроопределения с помощью команды **Play macro**. В дальнейшем для упрощения изложения будем говорить, что макроопределения связываются с функциональными клавишами, подразуме-

вая под ними и комбинацию функциональной клавиши **F10** и буквы.

Чтобы записать новое макроопределение, требуется нажать соответствующую функциональную клавишу (или комбинацию клавиши **F10** и буквы). Если эти клавиши уже заняты макроопределением, то нужно подтвердить решение о его перезаписи. После этого таблица и меню исчезают с экрана и можно вводить новое макроопределение, т.е. нажимать последовательность клавиш, которая войдет в состав макроопределения. При нажатии клавиш будут выполняться связанные с ними действия. Ввод макроопределения завершается клавишами **Shift-F10** и латинской буквой **e** или выбором позиции **End recording**.

После того как макроопределение создано, его можно модифицировать посредством добавления в его конец новой последовательности нажатий клавиш или изменением текста макроопределения. Первый способ реализуется выбором позиции **Append to macro**. Затем надо выбрать (нажать) соответствующую функциональную клавишу и приступить к дополнению макроопределения, как и при его создании, т.е. нажать клавиши в требуемой последовательности. Эта последовательность будет добавлена в конец уже имеющегося макроопределения. Второй способ модификации уже созданного макроопределения заключается в редактировании текста макроопределения с помощью текстового редактора. Этот способ реализуется выбором позиции **Modify**. Затем нужно нажать функциональную клавишу для высвечивания текста макроопределения. В этом тексте действия по нажатию буквенных клавиш записываются в виде букв, а специальных клавиш - в виде слов в фигурных скобках. Например, клавиши **Вверх**, **Вниз**, **Вправо** и **Влево** представляются соответственно в виде слов **{uparrow}**, **{downarrow}**, **{leftarrow}**, **{rightarrow}**. Допускается запись любыми буквами - строчными и прописными. Например, если требуется ввести в макроопределение вызов другого макроопределения, то вводится текст, в который входит команда выполнения макроопределения: **PLAY MACRO macro1{Enter}**, где **macro1** - имя макроопределения. В макроопределение можно ввести точку прерывания. Она дает возможность выполнять с клавиатуры различные операции, например вводить данные в файлы БД и др., после чего выполнение макроопределения возобновляется нажатием клавиш **Shift-F10**. Ввод точки прерывания осуществляется в ходе создания макроопределения. Для этого следует нажать клавиши **Shift-F10** и выбрать позицию **Insert user-input break**. Затем создание макроопределения продолжается и завершается, как обычно. Можно ввести несколько точек прерывания.

Макроопределение можно скопировать, связав с другой функциональной клавишей, посредством выбора позиции **Copy**. Затем следует нажать копируемую функциональную клавишу и ту клавишу, в которую копируется макроопределение. При

копировании можно изменить или оставить без изменения имя скопированного макроопределения. Уничтожение макроопределения выполняется выбором позиции **Delete** и последующим нажатием требуемой функциональной клавиши. Для пробного запуска созданного макроопределения можно использовать позицию **Play**, после выбора которой высвечивается таблица имен макроопределений. Запуск из этой таблицы осуществляется нажатием соответствующей функциональной клавиши. Макроопределение всегда выполняется с текущей позиции курсора. Если эта позиция отличается от позиции в момент создания макроопределения, то это может привести к результату, который не предполагался.

Для сохранения созданных макроопределений в текущей библиотеке выбирается позиция **Save library**. Если библиотека не создана, то высвечивается окно, в которое следует ввести имя библиотеки. Не сохраненные макроопределения после завершения текущего сеанса теряются. Загрузка ранее созданной библиотеки макроопределений осуществляется выбором позиции **Load library**. Если в момент загрузки имеется текущая библиотека, то высвечивается предупреждение для ее сохранения, которое можно сделать в этой же точке.

После сохранения или подтверждения не сохранять текущую библиотеку высвечивается список библиотек. После загрузки новой библиотеки ее макроопределения перекрывают старые, если они записаны под одним именем. В противном случае макроопределения старой библиотеки не заменяются и остаются действующими. Это позволяет комбинировать в оперативной памяти макроопределения из различных библиотек.

Макроопределение можно создать также из любого места пакета dBASE IV, вызвав сокращенное меню клавишами **Shift-F10**. После этого высвечивается горизонтальное меню с позициями **Begin recordering** и **Cancel**. Создание макроопределения осуществляется выбором первой из этих позиций. Вторая позиция позволяет отменить эту операцию.

Позиция **Import** меню **Tools** дает возможность переносить в dBASE IV внешние по отношению к нему файлы, присваивая им расширение **.dbf**. При этом поля записей сохраняют свои типы данных, созданные во внешних файлах. После выбора позиции **Import** высвечивается меню с опциями, реализующими выбор типа файла. Выбор типа файла приводит к высвечиванию списка файлов этого типа в текущем директории. Выбранный файл затем переносится в dBASE IV. Если такое имя уже есть в dBASE IV, то высвечивается предложение выполнить операцию или прекратить ее.

Можно импортировать следующие типы внешних файлов: **Rapidfile** (файлы с расширением **.rpd**), файлы БД **dBASE II** (для импорта этого типа файлов необходимо изменить их расширение с **.dbf** на **.db2**), **Framework II** (файлы БД и электронные таблицы

с расширением **.fw2**), **Lotus 1-2-3** (файлы с расширением **.wk1**, а для импорта файлов старших версий **Lotus 1-2-3** с расширением **.wks** следует использовать позицию **Copy records from non-dBASE file** из меню **Append**), **PFS:FILE** (файлы типа **PFS**, не имеющие расширения). Следует быть особенно внимательными при импорте последнего типа файлов. После ввода в dBASE IV им присаиваются расширения **.dbf**, **.vue**, **.fmt**.

Позиция **Export** меню **Tools** позволяет экспортировать файлы БД dBASE IV с расширением **.dbf** в другие типы внешних файлов. После создания внешнего файла ему автоматически присваивается соответствующее расширение. Можно экспортировать файлы БД dBASE IV в следующие типы файлов (в скобках указывается расширение файла): **Rapidfile** (**.rpd**), **dBASE II** (**.db2**) - перед использованием в dBASE II необходимо это расширение преобразовать в **.dbf**, **Framework II** (**.fw2**), **Lotus 1-2-3** (**.wks**), **VisiCalc** (**.dif**), **PFS:FILE** (без расширения), **Sylk-Multiplan** (без расширения), **Text fixed-length fields** (**.txt**) - файл типа SDF, **Blank Delimited** (**.txt**) - текстовый файл с разделителями в виде пробелов (можно задавать другие виды разделителей).

Позиция **DOS Utilities** меню **Tools** позволяет выполнять многие полезные операции при работе с файлами. После выбора этой позиции высвечивается таблица со списком имен файлов и их характеристиками, включая емкость занимаемой памяти в байтах (слово **<DIR>** обозначает имя директория), дату создания или последнего обновления, атрибуты файла (см. руководство по DOS) и память, занимаемую файлом на диске.

В двух нижних строках таблицы указывается количество и общий объем файлов в таблице, а также количество и общий объем помеченных файлов. Помеченные файлы составляют список, с которым можно выполнять операции как с одной группой, например удалять, копировать и т.д. По строкам таблицы можно передвигаться, используя навигационные клавиши. Для высвечивания списка файлов директория его следует выбрать (установить на него курсор и нажать **Enter**). Родительский (вышестоящий) директорий по отношению к текущему высвечивается выбором позиции **<parent>**. Повторный последовательный выбор этой позиции приводит к корневому директорию.

Движение по директориям легче выполнять по дереву директориев, которое высвечивается клавишей **F9** из любого места в таблице файлов. Передвижение по дереву также осуществляется навигационными клавишами. Высвечивание списка файлов директория выполняется выбором соответствующей позиции. Для перехода на другой диск нужно установить курсор на букву текущего диска в верхней части дерева и нажать клавишу **Enter** (высвечивается список имеющихся дисков). Последующий выбор диска приводит к высвечиванию дерева его директориев.

Позиция **DOS Utilities screen** включает меню с опциями: **DOS, Files, Sort, Mark, Operations** и **Exit**. Опция DOS позволяет выполнять команды DOS и устанавливать новый текущий диск и директорий. С помощью позиции **Perform DOS command** этой опции можно выполнять команды DOS непосредственно из dBASE IV. После завершения выполнения команды DOS для перехода к списку файлов требуется нажать любую клавишу.

Выбор позиции **Go to DOS** приводит к переходу в DOS. При этом высвечивается полный экран, на котором записываются команды DOS или программы исполнимых машинных команд. Эти команды и программы выполняются не выходя из dBASE IV. При этом метки файлов для выполнения групповых операций в таблице, из которой осуществлен выход в DOS, удаляются. Об этом дается специальное уведомление и предложение продолжить выполнение **(Proceed)** или отменить **(Cancel)** эти операции. После завершения операций DOS возврат в dBASE IV осуществляется вводом и выполнением команды **EXIT**.

Для установки диска и директория, которые будут использоваться во всех операциях по умолчанию (т.е. без явного указания диска и директория), используется позиция **Set default Drive: directory** (имя диска и директорий вводятся с клавиатуры в высвеченное окно). Можно выбрать директорий из списка директориев, высвечиваемого клавишами **Shift-F1**.

Позиция **Files** позволяет изменить диск и директорий файлов в таблице и высветить в ней только требуемые файлы. Смена диска и директория осуществляется выбором опции **Change Drive: directory**. Как и для установки диска, используемого по умолчанию, в высвечиваемое окно нужно ввести с клавиатуры требуемое имя диска и директория. Имя директория можно выбрать из списка, высвечиваемого клавишами **Shift-F1**. Для высвечивания в таблице файлов используется позиция **Display Only**, дающая возможность отобрать файлы с заданным расширением или одинаковой частью имени с помощью знаков замещения * и ?. Знак ? замещает одну позицию имени, а знак * - любое число позиций. Эту опцию можно использовать вместе с опцией **Mark** пометки групп файлов.

Позиция **Sort** служит для изменения порядка, в котором файлы представляются в списке. Текущий порядок показывается справа внизу таблицы. Может быть установлен порядок в соответствии с одним из перечисленных ниже способов, которые устанавливаются выбором следующих опций: **Name** (по имени файлов), **Extension** (по расширению файлов), **Date & Time** (по дате и времени создания файла), **Size** (по размеру файлов).

Позиция **Mark** дает возможность помечать файлы для групповой обработки. Файлы остаются помеченными до выхода в УЦ или в DOS командой **Go to DOS**. Различные операции по пометке файлов выполняются выбором опций: **Mark all** (пометка всех файлов в таблице), **Unmark all** (уничтожение всех пометок).

Reverse marks (реверс пометок - пометка неотмеченных файлов и удаление пометок ранее помеченных файлов).

Позиция **Operations** позволяет выполнять следующие операции с файлами: уничтожение, копирование, перемещение, переименование, просмотр, редактирование текстовых или программных файлов. При выполнении первых четырех операций высвечивается окно с позициями, позволяющими выбрать: один текущий файл (**Single**), группу отмеченных файлов (**Marked files**) или все файлы в таблице (**Displayed files**). Если курсор находится на имени директория, то при выборе позиции **Single** все операции за исключением переименования, просмотра и редактирования применяются ко всем файлам директория.

Уничтожение файла или группы помеченных файлов реализуется с помощью позиции **Delete**. Перед непосредственным уничтожением требуется подтвердить операцию (**Proceed**) или отменить ее (**Cancel**). Удалить один текущий файл можно также клавишей **Del**.

Опция **Copy** позволяет копировать помеченные файлы на другой диск и директорий, набираемый с клавиатуры в высвечиваемое окно. Если ранее уже выполнялось копирование, то высвечивается предыдущее назначение копирования, которое можно утвердить клавишей **Enter** или заменить новым, набранным с клавиатуры или выбранным из списка директориев, вызванным клавишами **Shift-F1**. Для копирования файлов с теми же именами, но в другой директорий в качестве имени нового файла (файла-мишени) можно использовать шаблон *.*. Нельзя копировать файл в самое себя. Копировать также можно клавишей **F8**. При этом высвечивается окно для ввода маршрута по директориям и имени файла-мишени.

Позиция **Move** осуществляет перемещение одного или группы помеченных файлов на новое место. Действия по выполнению операции аналогичны операции **Copy**. Переместить один текущий файл можно также клавишей **F7** (высвечивается окно для ввода маршрута по директориям и имени файла на новом месте). В этом случае операция должна завершаться клавишами **Ctrl-End**.

Для переименования файлов используется опция **Rename**. Просмотр содержимого файла осуществляется опцией **View**, при этом нетекстовые символы не высвечиваются. Файл можно просматривать страницами экрана, листая их клавишей **Пробел** или без остановки, нажав клавишу **Enter**. Выход из просмотра выполняется клавишей **Esc**.

Редактирование текстовых или программных файлов можно выполнить с помощью опции **Edit**.

Позиция **Protect data** служит для организации системы доступа в пакет dBASE IV в многопользовательском режиме и в среде ЛВС, а также для организации разграничения доступа и защиты информации. К этой позиции прикрепляется довольно

сложное иерархическое меню, содержание которого описано в разд. 2.11.

Позиция **Settings** меню **Tools** позволяет установить ряд параметров dBASE IV, например форму даты и разделитель позиций в ней, некоторые режимы обработки файлов, форматы полей в файлах БД, цвета различных областей экрана, а также сделать другие вспомогательные назначения. При выборе этой позиции высвечивается меню с позициями: **Options**, **Display** и **Exit**.

Позиция **Options** дает возможность выполнить в текущем сеансе установку на включение (**ON**) или выключение (**OFF**) значений параметров вывода и обработки данных. Эти параметры определяют подачу звукового сигнала, которая осуществляется при определенных ситуациях, например после заполнения поля данных (**Bell**), копирование данных в последующую запись из предыдущей (**Carry**), вывод в дате четырех цифр года (**Century**), автоматический переход курсора на следующее поле после заполнения предыдущего (**Confirm**), порядок вывода месяца, дня и года в дате (**Date order MDY**), разделитель в дате - "/", "-" или "." (**Date separator**), выводимое число десятичных позиций (**Decimal places**), пропуск помеченных для удаления записей при обработке файла БД (**Deleted**), требование точного совпадения искомой и заданной строк при их сравнении в ходе поиска (**Exact**), режим исключительного пользования файлом в среде ЛВС (**Exclusive**), высвечивание панели УЦ при входе в него (**Instruct**), сдвиг позиций слева при печати неформатированного текста (**Margin**), ширину печатаемого поля типа Memo (**Memo width**), предупреждение при перезаписи существующего файла (**Safety**), вывод на экран результатов выполнения операций (**Talk**), включение отладчика программ (**Trap**).

Позиция (меню) **Display** позволяет установить посредством выбора значений 25 или 43 в опции (позиции) **Display mode EGA** число строк на экране монитора типа **EGA**. В этом же меню можно назначить цвета различных областей экрана посредством выбора позиций: **Standard - All** - цвета всех элементов стандартных областей экрана, **Normal text** - цвет текста стандартной области экрана, **Messages** - цвет сообщений, **Titles** - цвет заголовков, **Enhanced - All** - цвет всех элементов нестандартных областей экрана, **Highlight** - цвет подсвечиваемых полей, **Boxes** - цвет рамок, **Information** - цвет информации в нестандартных областях экрана, **Fields** - цвет полей, **Perimeter of screen** - цвет периметра экрана.

Позиция **Exit** меню сервисных функций выполняет выход на панель УЦ.

В качестве примера работы с меню сервисных функций покажем более подробно, как установить требуемый маршрут по директориям, который будет нужен для просмотра учебного примера, европейский формат даты (типа ДД/ММ/ГГ, где

ДД-день, ММ-номер месяца, ГГ-год), а также как копировать, удалять, перемещать и переименовывать файлы. В предыдущем разделе описывалась операция копирования с помощью пакета Norton Commander, но для этого нужно выйти из dBASE IV в DOS. Здесь же будет описано, как выполнять эти операции средствами dBASE IV.

Для высвечивания файлов учебного примера на панели УЦ после входа в dBASE IV активизируйте меню сервисных функций клавишей **F10**, выберите подменю (позицию) **DOS Utilities** меню **Tools**. Затем активизируйте это подменю клавишей **F10** и выберите потом позицию **Set default drive:directory** меню (позиции) **DOS**. В результате в окне высветится текущий директорий, например, **D:\DBASEIV**. Установите в маршруте поддиректорий **PRIMER**, для чего добавьте наклонную черту влево и слово **PRIMER (D:\DBASEIV\PRIMER)** и нажмите клавишу **Enter**. Для выхода в УЦ нужно активизировать меню клавишей **F10**, перейти навигационной клавишей **"Вправо"** на позицию **Exit** и затем выбрать позицию **Exit to Control Center**. Можно использовать другой способ выхода, нажав клавишу **Esc** и затем подтвердить выход нажатием буквы **"Y"** на клавиатуре или выбором позиции **"YES"** в высветившемся меню. После этого на панели УЦ в окнах высветятся имена всех учебных файлов. Заметим, что при установке маршрута **D:\DBASEIV\PRIMER** предполагалось, что поддиректорий **PRIMER** директория **DBASEIV** уже создан (например, средствами пакета Norton Commander). Покажем теперь, как создать этот поддиректорий средствами dBASE IV. В меню **Tools** сервисных функций есть позиция, которая позволяет выполнять команды DOS не выходя из пакета dBASE IV. Для создания директория используется команда **mkdir** или сокращенно **md**. Чтобы создать поддиректорий **PRIMER** в директории **DBASEIV**, нужно, находясь в директории **D:\DBASEIV**, выбрать позицию **Perform DOS command** подменю второго уровня **DOS**, которое вызывается после выбора подменю первого уровня **DOS utilities** меню **Tools**. Заметим, что для активизации подменю **DOS** надо нажать клавишу **F10**. После этого высветится предложение ввести команду DOS. Введите с клавиатуры **md primer** и нажмите **Enter**. Поддиректорий создан. Ниже будет описано, как средствами dBASE IV копировать файлы.

Если Вы хотите решаемые примеры записывать на учебную дискету в директорий, где находятся учебные файлы, то после выбора позиции **Set default drive: directory** меню **DOS** укажите **A:\PRIMER** и нажмите клавишу **Enter**. Напомним, что в этом случае при решении примеров надо файлам давать имена, не совпадающие с именами на учебной дискете, иначе последние будут заменены Вашими примерами. Это же относится и к записи решаемых Вами примеров на жесткий диск в директорий учебных файлов.

Если Вы хотите записывать решаемые примеры не на жесткий диск или на учебную дискету, а на свою дискету, то после выбора позиции **Set default drive: directory** меню **DOS** сервисных функций укажите **A:** и нажмите клавишу **Enter**.

Для установки европейского формата даты выберите позицию **Settings** меню **Tools**, затем в высветившемся вертикальном меню, прикрепленном к позиции горизонтального меню **Options**, установите курсор на позицию **Date order** и клавишей **Пробел** выберите европейский формат даты - **DMY**. Перейдите в позицию **Exit** горизонтального меню и нажмите клавишу **Enter**. Установку формата даты можно также выполнить в командном режиме. Выход в командный режим осуществляется выбором позиции **Exit to dot prompt** меню **Exit** сервисных функций, которое активизируется клавишей **F10**. В командном режиме нужно набрать с клавиатуры команду **SET DATE GERMAN** и нажать клавишу **Enter**. Выход в УЦ из командного режима осуществляется набором с клавиатуры команды **ASSIST** или нажатием клавиши **F2**.

Операции по копированию, удалению, перемещению и переименованию файлов выполняются с помощью позиции **Operations** подменю **DOS utilities** меню **Tools**. После выбора подменю **DOS utilities** высвечивается окно с именами файлов диска и директория, в котором установлен пакет dBASE IV. В этом окне высвечиваются файлы, с которыми можно выполнять предусмотренные операции. Диск и директорий этих файлов можно установить, выбрав сначала позицию **Files** подменю **DOS utilities**, а затем в высветившемся меню позицию **Change drive: directory**. После выбора этой позиции курсор устанавливается в окно, где уже стоит имя диска и директория (директория, из которого запущен пакет dBASE IV или диска и директория предыдущей установки). После ввода имени нового диска и (или) директория нужно нажать клавишу **Enter**.

В этом меню имеется вторая позиция **Display only**, которая позволяет высветить в окне только те файлы, которые соответствуют задаваемому шаблону. Например, можно высветить только файлы БД или только файлы запросов. В первом случае в качестве шаблона нужно ввести ***.dbf**, а во втором случае ***.qbe**. Если требуется высветить только файлы БД, начинающиеся с буквы **F**, то надо ввести шаблон **F*.dbf**. Напомним, что знак ***** означает замещение любого количества символов. Кроме этого знака для замещения символов имени и расширения файла можно использовать один или несколько знаков **?**, каждый из которых замещает только один символ. Например, шаблон **MQ??.qbe** позволяет высветить файлы запросов, имена которых начинаются с букв **MQ** и всего содержат четыре символа. После ввода шаблона также надо нажать клавишу **Enter**.

Заметим, что директорий, файлы которого высвечиваются в окне, и директорий, в который записываются файлы по

умолчанию, т.е. без специального указания диска и маршрута (часто говорят, что это директорий, установленный по умолчанию), - это в принципе разные директории.

Первый используется только для высвечивания в окне файлов, с которыми можно выполнять операции, а второй - это директорий, с которым по умолчанию, выполняются операции: считываются или записываются файлы. После входа в dBASE IV по умолчанию автоматически устанавливается директорий, из которого он вызван. Как уже упоминалось, смена диска и директория, установленного по умолчанию, осуществляется с помощью позиции **Set default drive:directory** меню (позиции) **DOS** подменю **DOS Utilities** меню **Tools**.

Можно, например, выполнять операции с каким-то директорием жесткого диска, затем прервать эту работу и выполнить операции по копированию на жесткий диск файлов с дискеты или, наоборот, скопировать файлы с жесткого диска на дискету. Для этого надо установить имя и директорий копируемых файлов (например, для дискеты **A:\PRIMER**, а для жесткого диска **C:\DBASEIV\PRIMER**) с помощью позиции **Change drive: directory** подменю **Files**, которое вызывается из подменю **DOS utilities** меню **Tools**. (Напомним, что имена дисков, директориев и файлов можно вводить как строчными, так и прописными латинскими буквами.) После завершения операции копирования можно продолжить работу с диском и директорием, установленным ранее по умолчанию, который остается без изменения.

После того как установлен требуемый диск и директорий, с файлами которого нужно выполнить операции, следует выбрать позицию **Operations** подменю **DOS utilities** меню **Tools**. Напомним, что после выбора подменю **DOS utilities** и высвечивания окна с файлами, меню следующего уровня находится в неактивном состоянии. Для активизации меню нужно нажать клавишу **F10**. После выбора позиции **Operations** высвечивается меню, в котором перечислены операции: удалить (**Delete**), копировать (**Copy**), переместить (**Move**), переименовать (**Rename**), просмотреть (**View**) и редактировать (**Edit**). Если нужно выполнить операцию с одним файлом, достаточно установить на его имя курсор. Перед выполнением операций копирования, удаления или перемещения нескольких файлов их имена надо пометить, что эквивалентно подсвечиванию файлов при использовании пакета **Norton Commander**. Пометка файлов по-одному выполняется клавишей **Enter** после установки курсора на имя файла. Знаком пометки служит треугольник, помещаемый слева от имени файла.

Все файлы директория можно пометить с помощью позиции **Mark** подменю **DOS utilities**. После выбора этой позиции высвечивается меню с позициями: **Mark all** - пометить все файлы, **Unmark all** - погасить пометки всех файлов и **Revers** - погасить имеющиеся пометки и пометить ранее не помеченные

файлы. Если требуется выполнить операцию со всеми высвеченными файлами, то их можно не помечать. После того, как файлы помечены или, если операция выполняется с одним файлом, на его имя установлен курсор, можно выполнять требуемые операции. Для этого следует выбрать требуемую операцию из меню, высвечиваемого после выбора позиции **Operations**. После этого высвечивается меню с тремя позициями **Single File**, **Marked Files** и **Displayed Files**, одну из которых нужно выбрать в соответствии с тем, как определены файлы, с которыми требуется выполнить эти операции. Выбор первой позиции предполагает выполнение операции с одним файлом, на который установлен курсор. Выбор второй позиции предполагает выполнение операции с помеченными файлами, а выбор третьей предполагает выполнение операции со всеми высвеченными в окне файлами.

При выполнении операции по удалению файлов высвечивается сообщение, что выполняется эта операция и меню с двумя позициями, в котором для подтверждения удаления нужно выбрать позицию **Proceed**, а для отмены удаления позицию **Cancel**. Для операций копирования и перемещения нужно ввести с клавиатуры имя диска и директорий, куда копируются или перемещаются файлы, и нажать **Enter**, затем можно изменить старое имя файла, которое высвечивается в окне, на новое и нажать клавиши **Ctrl-End** для завершения операции. При переименовании нужно ввести новое имя. Просмотр и редактирование файлов выполняется только по-одному. Заметим, что эти операции можно использовать только с текстовыми файлами, причем в неадаптированной версии dBASE IV при просмотре файла кириллица выводится искаженно.

2.4. СОЗДАНИЕ ФАЙЛОВ БАЗ ДАННЫХ И ВВОД ЗАПИСЕЙ

Рассмотрим теперь, как создать структуру файла БД. Для этого надо выбрать позицию **<create>** окна **Data** на панели УЦ. Вспомним, что мы условились понимать под выбором позиции: установить курсор на нужной позиции и нажать клавишу **Enter**. На экране появится таблица, состоящая из шести колонок с именами: порядковый номер поля (**Num**), имя поля (**Field Name**), тип поля (**Field Type**), ширина поля (**Width**), количество дробных десятичных знаков числа (**Dec**), признак индексирования поля (**Index**). Каждая строка этой таблицы содержит информацию об одном поле записи файла БД. Последовательность номеров полей соответствует их расположению в записи файла БД. До тех пор, пока структура файла БД не записана в каталог, ее можно свободно изменять посредством передвижения курсора и изменения описаний полей данных клавишами ввода и удаления символов. Имя поля может включать до **10** бук-

венно-цифровых символов, но должно начинаться обязательно с буквы. Тип поля (символьный - **Character**, числовой с фиксированной точкой - **Numeric**, числовой с плавающей точкой - **Float**, текстовый - **Memo**, логический - **Logical** и типа даты - **Date**) можно выбрать клавишей **Пробел** или набрать непосредственно с клавиатуры. Ширина поля для символьных полей ограничена **254** позициями, а для числовых полей **20** позициями (включая знак числа, десятичную точку и дробные десятичные знаки). Ширина полей типа даты и логических полей фиксирована и вводится в таблицу автоматически. Данные из текстового поля **Memo** (шириной до **32767** символов) хранятся не в самом файле БД, а в связанном с ним специальном файле. В таблицу описания полей при создании поля **Memo** автоматически вводится фиксированная ширина ссылки (**10** позиций) на этот файл. Завершается ввод информации по одному полю указанием признака для создания индекса по этому полю (**Y** - индекс создается, **N** - не создается).

Для решения задачи учета работы автотранспортных средств сформируем структуры четырех файлов БД. В учебных примерах им присвоены имена **BD1-BD4**. Чтобы не было совпадения имен файлов при работе с учебной дискетой, условимся в дальнейшем давать имена создаваемым Вами файлам те же, что и в учебных примерах, но с буквой **M** впереди. Тогда для файлов БД эти имена будут **MBD1-MBD4**. Заметим, что имена полей нужно вводить в таблицу структуры БД с первой позиции. Движение по полям таблицы вправо осуществляется клавишей **Tab**, а влево клавишами **Shift-Tab**. (Здесь и далее запись английских имен клавиш через тире означает, что первую клавишу нужно держать нажатой, а вторую нажимать после для вызова действия.) Сначала создадим структуру файла БД **MBD1** (табл. 2.3).

После того как введено описание для всех полей файла БД, созданную структуру следует сохранить на диске. Для этого надо сначала клавишей **F10** вызвать меню экрана создания/модификации структуры БД и затем выбрать позицию **Save this database file structure** (сохранить структуру файла БД) меню **Layout**.

Таблица 2.3. **Структура файла MBD1**

Num	Field Name	Field Type	Width	Dec	Index
1	NOM	Numeric	4	0	Y
2	FAM	Character	15		Y
3	NAM	Character	15		N
4	OTC	Character	15		N
5	D_R	Date	8		N
6	S_R	Numeric	2	0	N
7	KAT	Character	9		N
8	AUTO	Memo	10		N
9	N_Z	Character	9		Y

В ответ на запрос **Save as:** вводится имя диска, маршрут по директориям и в конце имя файла БД по форме **D:\DBASEIV\PRIMER\MBD1**. Сохранение структуры созданной БД можно также выполнить выбором позиции **Save changes and exit** меню **Exit** с последующим выходом в УЦ.

Для удобства работы в УЦ с созданным файлом БД можно ввести его словесное описание, выбрав позицию **Edit database description** (редактировать описание файла БД) этого же меню. Напомним, что меню всегда вызывается клавишей **F10**. Введем описание файла **BD1: "Данные о водителях"**. Можно также распечатать созданную структуру, выбрав позицию **Print database structure**.

Выход из экрана создания/модификации файлов БД без сохранения осуществляется выбором позиции **Abandon changes and exit** меню **Exit**.

Выход без сохранения также можно выполнить клавишей **Esc** с последующим подтверждением **"Y"**.

Для закрепления описанных действий по созданию структуры БД самостоятельно создайте структуры файлов БД с именами **MBD2**, **MBD3**, **MBD4**, например, так, как показано в табл. 2.4-2.6.

Если после создания структуры файла БД обнаружены какие-либо ошибки, то структуру можно изменить (модифицировать). Напомним, что перед выполнением любых действий с файлом БД его следует сначала открыть, т.е. выбрать имя файла в окне **Data** на панели УЦ. Вспомним, что значит выбрать имя файла: установить курсор на его имя в окне и нажать **Enter**. После этого появится горизонтальное меню, в котором нужно выбрать позицию **Use file** (открыть файл). Если требуется файл модифицировать, то после его выбора в окне **Data** на панели УЦ, затем в высветившемся окне нужно выбрать позицию **Modify structure/order** (изменить структуру/порядок) горизонтального меню.

Модификация осуществляется так же, как и создание новой структуры файла БД. При модификации можно удалять и вводить новые поля и изменять их размеры и тип. Перед вводом нового поля в требуемое место таблицы нужно нажать клавиши **Ctrl-N**, в результате чего в этом месте таблицы появится пустая строка, в которую можно вводить данные. Удаление любой строки таблицы структуры файла БД выполняется клавишами **Ctrl-U**.

Таблица 2.4. **Структура файла MBD2**

Num	Field Name	Field Type	Width	Dec	Index
1	N_Z	Character	9		Y
2	SOS	Character	12		N
3	MAR	Character	10		Y
4	G_N	Character	12		N

Таблица 2.5. **Структура файла MBD3**

Num	Field Name	Field Type	Width	Dec	Index
1	N_G	Character	20		N
2	O_P	Character	15		N
3	D_V	Date	8		N
4	K_D	Character	15		N
5	D_P	Date	8		N
6	RAS	Numeric	4	0	N
7	VES	Numeric	4	1	N
8	VRM	Numeric	3	0	N
9	RGR	Numeric	6	2	N
10	N_Z	Character	9		Y

Таблица 2.6. **Структура файла MBD4**

Num	Field Name	Field Type	Width	Dec	Index
1	MAR	Character	10		Y
2	SKR	Numeric	3	0	N
3	GRZ	Numeric	2	0	N
4	RSX	Numeric	2	0	N

Следует только учесть, что при изменении размера полей в меньшую сторону данные могут быть обрезаны, а при изменении типа полностью потеряны. Нужно отметить также, что при модификации нельзя изменить порядок расположения полей. Это можно сделать другим способом, например, скопировав этот файл в файл с другим именем и такой же структурой, но с другим расположением полей.

Продолжим обзор меню окна **Data** панели УЦ. Кроме меню (позиции) **Layout** в состав основного (горизонтального) меню, рассмотренного выше, входит еще меню (позиция) **Organize**, предназначенное для создания индексных файлов, а также некоторых других операций с файлами БД, меню (позиция) **Append** - для выполнения операций по добавлению записей в файл БД, меню (позиция) **Go To** - для передвижения по записям файла БД и меню (позиция) **Exit** - для выхода из меню.

Как уже отмечалось, индексные файлы создаются для более быстрого поиска данных и вывода записей в определенном порядке без физической сортировки файла, при которой создается новый файл БД. Для этого требуется довольно много времени и значительная емкость памяти на диске, равная удвоенному объему исходного файла. При использовании индексов создается сравнительно небольшой индексный файл, который используется многократно в процессе поиска или вывода данных из файла БД на экран или печать. Перед поиском необходимо этот файл активизировать - открыть и установить

главный (управляющий) индекс, который и будет обеспечивать операции поиска и вывода данных.

Как уже упоминалось, в dBASE IV все индексы сводятся в один множественный индексный файл типа **.mdx** (иногда их называют указателями индексов). Индексы могут строиться по отдельным полям (простые) или комбинациям полей (сложные). Всего можно создать до **47** указателей индексов в одном множественном индексном файле.

Простые индексы создаются автоматически после окончания разработки структуры файла БД (при выборе **Y** в колонке **Index** таблицы структуры файла). Образованный этим способом множественный индексный файл называется связанным, так как он создается автоматически и ему присваивается имя файла БД. При формировании сложных индексов (указателей индексов) создается множественный индексный файл, которому необходимо присвоить уникальное имя. Таких множественных индексных файлов можно создать до 10 на один файл БД.

При формировании структур файлов БД **BD1**, **BD2**, **BD3**, **BD4** были созданы множественные индексные файлы: **BD1.mdx**, содержащий индексы полей **NOM**, **FAM**, **N_Z**; **BD2.mdx** - с индексами полей **N_Z**, **MAR**; **BD3.mdx** - с индексом поля **N_Z**; **BD4.mdx** - с индексом поля **MAR**.

Для совместимости с dBASE III+ в пакете dBASE IV оставлена возможность создавать и использовать индексные файлы типа **.ndx**. В текущий момент управляющим может быть только один индексный файл или указатель индекса в множественном индексном файле (главный индекс). Меню **Organize** предназначено для того, чтобы создавать и изменять сложные индексы, использовать индексы типа **.ndx**, активизировать индексы и выполнять другие операции, связанные с индексными файлами, например, удалять ненужные индексы, а также сортировать файл БД, уничтожать помеченные для удаления записи и другие операции.

Выше описывался способ автоматического создание индекса в ходе формирования структуры файла БД. Кроме этого способа можно создать индекс с помощью позиции **Create new index** меню **Organize**. Обычно этот способ используется для создания сложных индексов, которые нельзя построить первым способом. После выбора этой позиции высвечивается окно с позициями для присвоения имени этому индексу (**Name of index**), создания выражения, определяющего сложный индекс (**Index expression**), определения порядка индексирования по возрастанию или убыванию (**Order of index**). Имя индекса формируется по правилам для имен файлов: до **10** буквенно-цифровых символов, начиная с буквы. Выражение для сложного индекса состоит из комбинации полей с операторами и функциями, разрешенными для этих полей. Например, для символьных полей разрешена операция сцепления и сложный индекс может состоять из

последовательности имен полей, соединенных знаком "+" (знак сцепления полей). Имена полей можно набрать с клавиатуры или выбрать из списка, высвечиваемого клавишами **Shift-F1**. Порядок индексирования определяется выбором клавишей **Пробел** слова **Asc** для индексирования по возрастанию и **Desc** - по убыванию. Если несколько записей имеют одно и то же значение индексного выражения, то для вывода (на печать или экран) только одной (первой) такой записи используется позиция **Display first duplicate key only**. После ввода всех необходимых данных в позиции окна их следует сохранить (записать на диск) клавишами **Ctrl-End**.

Для изменения уже имеющихся индексов применяется позиция **Modify existing index**, а для установки главного индекса - позиция **Order records by index**. Важным преимуществом множественного индексного файла является то, что при установке главного индекса не нужно открывать новый индексный файл. В случае использования файла типа **.ndx** его необходимо предварительно открыть, что занимает определенное время. Эта операция выполняется выбором позиции **Activate .ndx index file**. Для включения в текущий каталог файла типа **.ndx**, находящегося в текущем директории (при частом употреблении), используется позиция **Include .ndx index file**.

При модификации задачи некоторые индексы могут оказаться ненужными. Поскольку для больших файлов они занимают много памяти на диске, целесообразно их удалить. Для этого используется позиция **Remove unwanted Tag**.

Созданный файл можно физически отсортировать по заданным полям в возрастающем или убывающем порядке с помощью позиции **Sort database on field list**. Для того чтобы отсортировать файл БД, необходимо сначала его открыть (выбрать на панели УЦ в окне **Data**). После этого клавишами **Shift-F2** нужно перейти в меню **Organize** и выбрать позицию **Sort database on field list**. В результате высветится таблица ввода данных для сортировки с двумя графами: **Field order** (имена сортируемых полей) и **Type of sort** (тип сортировки).

В первую графу заносятся имена нескольких полей (или одно поле) в том порядке, который определяет последовательность сортировки. Следует отметить, что сортировка может выполняться только по индексированным полям. Имена полей вводятся с клавиатуры или из списка, высвечиваемого клавишами **Shift-F1**.

Во вторую графу вводится тип сортировки: **Ascending** - по возрастанию, **Descending** - по убыванию, **Ascending Dictionary** - по возрастанию без различия прописных и строчных букв, **Descending Dictionary** - аналогично по убыванию. Заданные значения надо сохранить клавишами **Ctrl-End**, затем ввести имя отсортированного файла и нажать клавишу **Enter** для выполнения сортировки.

Это же меню позволяет уничтожить все имеющиеся пометки для удаления записей в файле БД (**Unmark all records**), а также помеченные для удаления записи (**Erase marked records**).

Меню **Append** дает возможность добавлять в созданный файл БД записи с клавиатуры (**Enter records from keyboard**), из другого файла БД (**Append records from dBASE file**) и из внешнего (не-dBASE) файла (**Copy records from non-dBASE file**). При добавлении с клавиатуры открывается стандартная экранная форма **Browse** или **Edit**, описание которых дается ниже. В случае добавления записей из другого файла имя этого файла нужно выбрать из списка имен файлов БД в текущем или других директориях в высвечиваемом на экране списке. После выбора имени файла выполняется добавление записей из этого файла БД.

При добавлении записей из внешних файлов высвечивается таблица с именами типов внешних файлов (в скобках указывается расширение файла): **Rapidfile (.rpd)**, **dBASE II (.db2)** - перед использованием в dBASE II необходимо это расширение преобразовать в **.dbf**, **Framework II (.fw2)**, **Lotus 1-2-3 (.wks)**, **VisiCalc (.dif)**, **PFS:FILE** (без расширения), **Sylk-Multiplan** (без расширения), **Text fixed-length fields (.txt)** - текстовой файл типа SDF, **Blank Delimited** (.txt) - текстовой файл с разделителями в виде пробелов (можно задавать другие виды разделителей).

После выбора требуемого типа файла высвечивается список директориев и имен файлов этого типа на текущем диске. Для того чтобы перейти в директорий высшего уровня, нужно выбрать позицию **<parent>** этого списка. Другие директории также раскрываются их выбором. Если требуется перейти на другой диск, то курсор надо установить на позицию с именем текущего диска и нажать клавишу **Enter**. После выбора имени внешнего файла выполняется добавление его записей в конец файла БД dBASE IV. При добавлении записей следует учитывать соответствие длины и типа полей внешнего и принимающего файлов. При несоответствии данные могут усекаться или восприниматься не в том виде, как это предполагалось.

Ввод новых и обновление существующих записей, как и их добавление, выполняется в одних и тех же стандартных экранных формах **Browse** и **Edit**. Напомним, что предварительно нужно файл БД открыть. Факт открытия файла отмечается на панели УЦ тем, что его имя перемещается выше черты под слово **<create>**. Выход на экраны ввода и обновления данных можно выполнить несколькими способами. Находясь на экране разработки структуры файла БД, после ее окончания можно сразу перейти к вводу данных посредством выбора позиции **Enter records from keyboard** меню **Append**.

Кроме того, вход в стандартные экранные формы можно выполнить из УЦ. Для этого после установки курсора на имя файла в окне **Data** нужно нажать клавишу **Enter** и затем выбрать

позицию **Display data** или более коротким способом - нажать клавишу **F2**. После этого высветится одна из стандартных экранных форм: **Browse** или **Edit** (в зависимости от того, какая из форм была на экране в предыдущем сеансе обновления, переключение на альтернативную форму осуществляется клавишей **F2**). Следует отметить, что такое переключение возможно только, если файл не пустой. Поэтому сначала надо ввести данные хотя бы в одну запись в высвеченной форме, а затем установить желаемую форму.

В форме **Browse** данные высвечиваются в табличном виде (строка таблицы - запись файла), при этом не все поля могут одновременно разместиться на экране. Для просмотра скрытых полей и ввода в них информации следует использовать навигационные клавиши **Tab** и **Shift-Tab** (для движения по полям соответственно в прямом и обратном направлениях), клавиши **Home** и **End** для быстрого передвижения в левую и правую части таблицы, клавиши со стрелками **Вправо** и **Влево** - для перемещения внутри полей, клавиши со стрелкам **Вверх, Вниз** и **PgUp, PgDn** - для перехода по строкам и страницам таблицы. Информация, вводимая в поля файла БД, сначала записывается в промежуточный буфер, поэтому до момента перенесения ее на диск еще можно восстановить старые данные.

Перенос информации на диск осуществляется автоматически после перехода к следующей записи или после выхода из операции ввода. Следует отметить, что данные последней введенной записи не сохраняются на диске, если выход из операции ввода осуществляется с помощью клавиши **Esc** или другим некорректным способом. Во избежание потери данных выход из операции ввода следует выполнять только с помощью позиции **Exit** меню **Exit** или нажатием клавиш **Ctrl-End**.

В формах **Browse** и **Edit** все поля, за исключением поля типа **Memo**, представлены явно, и в них можно непосредственно вводить или обновлять данные. Поле типа **Memo** представляется на экране в виде маркера - слова **Memo**. Если это слово записано строчными буквами, то это поле не заполнено информацией, если прописными, то информация в этом поле уже есть. Раскрытие поля для ввода или просмотра данных в нем выполняется клавишами **Ctrl-Home** или клавишей **F9**, а закрытие - клавишами **Ctrl-End** или **F9**.

В форме **Edit** поля одной записи размещаются одно под другим на одной или нескольких страницах экрана (в зависимости от количества полей). Раскрытие последующих страниц выполняется клавишей **PgDn**. Передвижение по полям и записям осуществляется так же, как и в форме **Browse**.

Экранные формы **Browse** и **Edit** имеют свои меню для выполнения многих операций при вводе и обновлении данных файла БД. Эти операции можно разбить на четыре группы, которые имеют свои меню: операции, обеспечивающие работу

с записями файла БД (**Records**), с полями записи (**Fields** - только для экранной формы **Browse**), передвижение по файлу БД, поиск в нем данных (**Go To**) и выход из меню экранной формы (**Exit**). Содержание позиций этих меню приводится ниже.

Следует отметить, что в современных СУБД, включая dBASE IV, с целью защиты от неосмотрительных действий отсутствуют непосредственные команды для удаления записей. Вместо прямого удаления используется предварительная пометка (маркировка) записи на удаление. Собственно удаление группы помеченных записей выполняется отдельно с помощью специальной команды. Пометку текущей записи можно осуществить клавишами **Ctrl-U** или с помощью позиции **Mark record for deletion** меню **Records** экранной формы. Помеченная для удаления запись отмечается индикатором **Del**, который высвечивается в правой части строки состояния, находящейся в нижней части экрана. Вместо удаления записи рекомендуется использовать позиции меню экранных форм **Browse** и **Edit** для удаления содержимого (заполнения пробелами) целой записи (позиция **Blank record** меню **Records**) или отдельных полей (позиция **Blank field** меню **Fields** - только для формы **Browse**). Кроме этого, можно использовать клавиши **Ctrl-Y** для удаления текста поля, расположенного вправо от текущей позиции.

Собственно удаление помеченных записей выполняется в экране создания/модификации структуры файла БД (вход в этот экран осуществляется из УЦ установкой курсора на имя файла БД в окне **Data** и последующим нажатием клавиш **Shift-F2**) выбором позиции **Erase marked records** меню **Organize**. Напомним, что горизонтальное меню активизируется клавишей **F10**. После этого навигационными клавишами нужно перейти на позицию **Organize** и нажать клавишу **Enter**, в результате чего раскроется прикрепленное вертикальное меню, в котором выбирается нужная позиция.

Вообще помеченные для удаления записи часто не требуется удалять. Пометка записей позволяет временно скрыть их - сделать невидимыми для поиска и отбора данных. После того как надобность в этом отпадет, помеченные записи можно размаркировать с помощью позиции **Unmark all records** меню **Organize** в экране создания/модификации структуры файла БД. Размаркировка текущей записи выполняется в форме **Edit** или **Browse** нажатием клавиш **Ctrl-U** или выбором позиции **Clear deletion mark** меню **Records**.

Для предотвращения потерь информации предусмотрена команда восстановления содержимого всей записи после выполнения изменений (позиция **Undo change to record**). Добавление новой записи можно осуществить двумя способами: если последняя запись является текущей, то навигационной клавишей **Вниз** (внизу экрана высвечивается сообщение **Add new**

records ? (Y/N) - ответить **Y**). Если текущая запись не последняя, то выбором позиции **Add new records** меню **Records**.

В форме **Browse** можно зафиксировать ряд полей на экране слева при передвижении курсора вправо (позиция **Lock fields on left** меню **Fields**), очистить поле - ввести пробелы (позиция **Blank field** меню **Fields**), осуществить захват записи при работе в среде ЛВС (**Lock record**), сделать выбранное поле доступным для редактирования, а остальные закрыть для обновления (позиция **Freeze field** меню **Fields**), изменить размер колонки высвечиваемого поля на экране (позиция **Size field** меню **Fields**).

Для передвижения по записям файла БД и поиска данных в формах **Browse** и **Edit** используется меню **Go To**, которое включает следующие позиции: переход к первой записи (**Top record**), к последней записи этого файла (**Last record**), к записи по номеру (**Record number**), пропуск **N** записей (**Skip**) (**+N** - вперед, **-N** - назад), поиск с использованием индекса поля (**Index key search**), поиск вперед (**Forward search**), поиск назад (**Backward search**), установка требования различать (**Yes**) или не различать (**No**) при поиске прописные и строчные буквы (**Match capitalization**).

При выходе из меню экранной формы **Browse** или **Edit** с помощью меню **Exit** можно передать управление на панель УЦ (позиция **Exit**) или в меню разработки запросов на отбор данных (позиция **Transfer to query designe**).

Проиллюстрируем, как ввести данные в файл БД. В качестве упражнения введите в файлы **MBD1**, **MBD2**, **MBD3** и **MBD4** данные, приведенные в табл. 2.7-2.10. Заметим, что данные в поле **KAT** (категория водительских прав) введены латинскими буквами.

Первоначальный ввод данных представляет собой важный этап при создании БД. Обычно при этом требуется ввести большой объем информации. Поскольку БД чаще всего составляет систему связанных файлов, то часть информации неизбежно повторяется. Для того, чтобы уменьшить объем первоначально вводимых данных, нужно продумать последовательность этапов ввода и использовать предоставляемые УЦ возможности по автоматизации некоторых процессов. В нашей задаче объем первоначально вводимых данных можно сократить следующим образом. Сначала введите данные о водителях в файл **MBD1**, затем, скопируйте номерные знаки из файла **MBD1** в файл **MBD2**. После этого введите в файл **MBD2** недостающую информацию.

Копирование номерных знаков из поля **N_Z** файла **MBD1** в файл **MBD2** можно осуществить посредством операции добавления записей в файл **MBD2** из файла **MBD1**. При этом будут добавляться только данные одноименных полей. Для этого после ввода всей информации в файл **MBD1** в окне **Data** УЦ выберите файл **MBD2** и перейдите на экран модифицирования его структуры (клавишами **Shift-F2**).

Таблица 2.7. **Файл MBD1**

NOM	FAM NAM OTC	D_R	S_R	KAT	AUTO	N_Z
1	Архипов Сергей Иванович	23/03/49	10	A,C	Родился 23 марта 1949 г. в Москве	34-59 ММС
2	Бронзов Сергей Васильевич	12/11/60	5	B,C	Родился 12 ноября 1960 г. в Харькове	09-33 ММД
3	Безродный Владимир Михайлович	05/09/53	7	A,B C,E	Родился 5 сентября 1953 г в Москве	99-38 ММС
4	Бородулин Алексей Владимирович	31/12/60	2	C	Родился 31 декабря 1960 г. в Киеве	01-01 ММД
5	Ермишкин Юрий Иванович	06/05/45	11	C,E A	Родился 6 июня 1945 года в Москве	78-99 ММС
6	Малонушенко Юрий Николаевич	21/11/64	6	A,B C	Родился 21 ноября 1964 г. в Омске	35-68 ММД
7	Поляков Геннадий Михайлович	17/07/39	19	B,C E	Родился 17 июля 1939 в Москве	43-92 ММС
8	Коваль Александр Николаевич	30/03/65	3	A,C	Родился 30 июня 1965 г. в Киеве	59-31 ММД
9	Садчиков Андрей Викторович	13/01/57	8	C	Родился 5 августа 1965 г. в Тамбове	99-99 ММД
10	Строков Олег Валерьевич	05/08/65	4	B,C	Родился в Зеленограде в 1965 г.	34-56 ММД
11	Старченко Сергей Николаевич	22/04/43	10	C	Родился в Москве 22 марта 1943 г.	01-34 ММД

Таблица 2.8. **Файл MBD2**

N_Z	SOS	MAR	G_N
34-59 ММС	Не исправен	ЗИЛ-131	В автопарке
09-33 ММД	Исправен	ГАЗ-53	В автопарке
99-38 ММС	Исправен	ГАЗ-53	В рейсе
01-01 ММД	Исправен	ЗИЛ-131	В рейсе
78-99 ММС	Исправен	ЗИЛ-131	В рейсе
35-68 ММД	Не исправен	ЗИЛ-131	В капремонте
43-92 ММС	Исправен	ЗИЛ-130	В рейсе
59-31 ММД	Исправен	ГАЗ-53	В рейсе
99-99 ММД	Исправен	ЗИЛ-131	В рейсе
34-56 ММД	Исправен	ГАЗ-53	В автопарке
01-34 ММД	Исправен	ГАЗ-53	В рейсе

Таблица 2.9. **Файл MBD3**

N_G	O_P K_D	D_V D_P	RAS	VRM	VES	RGR	N_Z
Станки ЧПУ	Одинцово Горький	24/03/89 25/03/89	300	14	3.0	44.00	34-56 ММД
Стройматериалы	Москва Ярославль	22/05/89 20/05/89	300	13	4.0	50.00	78-99 ММС
Мебель	Москва Калуга	30/06/89 01/07/89	300	12	1.0	46.00	34-56 ММД
Мебель	Москва Горький	25/07/89 26/07/89	400	16	3.0	69.00	99-99 ММД
Стройматериалы	Болшево Тула	10/10/89 12/10/89	200	8	2.0	33.00	78-99 ММС
Техника	Москва Подольск	15/10/89 15/10/89	40	2	3.0	8.00	99-99 ММД
Двигатели	Москва Рязань	12/10/89 14/10/89	600	22	3.0	72.00	34-56 ММД

Таблица 2.10. **Файл MBD4**

MAR	SKR	GRZ	RSX
ГАЗ-53	70	3	14
ЗИЛ-130	80	5	15
ЗИЛ-131	75	6	17

Затем в меню **Append** установите курсор на позицию **Append records from dBASE file** (добавить записи из dBASE-файла) и из высветившегося вертикального меню выберите файл **MBD1**. В результате выполнения операции все данные, содержащиеся в поле **N_Z** файла **MBD1**, будут введены в поле **N_Z** файла **MBD2**.

Следует отметить, что ввод текста с клавиатуры можно выполнять в двух режимах. В режиме **Ins** (нажата клавиша **Ins**, что отмечается словом **Ins** в правой части нижней информационной строки экрана) символы вводятся с раздвижкой всего текста вправо. В альтернативном режиме (переключается повторным нажатием клавиши **Ins**) символы вводятся по имеющемуся на экране тексту. Режим **Ins** целесообразно применять при вставке слов и предложений в ранее введенный текст, а альтернативный режим при замене этого текста. В ряде случаев объем ручной работы по вводу информации можно сократить посредством копирования данных из поля предыдущей записи в это же поле последующей. Для этого надо установить курсор в поле последующей записи и нажать клавиши **Shift-F8**.

Объем работы при добавлении новых записей можно сократить, установив режим автоматического копирования данных

всех полей последней записи в новую добавляемую запись с помощью команды **SET CARRY ON** в командном режиме или с помощью меню **Tools** сервисных функций. В последнем случае надо сначала выбрать позицию **Settings** этого меню, затем в подменю **Options** установить значение **ON** опции **Carry**.

Добавление новых записей по-одной можно выполнить с помощью позиции **Add new records** меню **Browse** или **Edit** или просто нажатием клавиши **Вниз**, находясь на последней записи. В последнем случае в нижней строке экрана высвечивается сообщение **Add new records ? (Y/N)**. Для ввода нужно нажать клавишу **Y**, а при отказе - **N** (перед нажатием клавиш **Y** или **N** следите за тем, чтобы клавиатура была переключена на режим ввода латинских букв). При установленном режиме **SET CARRY ON** все поля последней записи копируются в добавленную запись. Следует отметить, что если никаких изменений в последней скопированной записи не делать, то для добавления в файл копии предыдущей записи нужно нажать клавишу **Вниз** дважды.

Чтобы не выполнять рутинную работу по вводу данных в созданные Вами файлы БД, можно скопировать эти данные из эталонных учебных файлов с именами **BD1-BD4**. Эта операция выполняется так же, как и при копировании поля **N_Z** из файла **MBD1** в файл **MBD2** с той разницей, что теперь надо добавить записи из эталонных учебных файлов **BD1-BD4** в созданные Вами файлы **MBD1-MBD4**. Поскольку имена полей одинаковые, то все данные из эталонных учебных файлов будут переписываться в Ваши файлы.

2.5. ЗАПРОСЫ НА ПОИСК И ОТБОР ДАННЫХ

После того как создана структура файла БД и введена информация, обычно требуется отбирать и выводить на экран или на печать те данные, которые отвечают поставленной задаче. Для этого формируются запросы на отбор необходимых данных. Запросы на отбор позволяют формировать так называемые выборки. Выборка представляет собой данные из одного или нескольких файлов БД, которые не существуют в виде самостоятельного файла, а только высвечиваются на экране как результат выполнения запроса.

Кроме отбора данных можно формировать запросы с целью обновления данных. Выше был рассмотрен ручной способ ввода и обновления данных в экранных формах **Browse** и **Edit**. Обновление с помощью запросов позволяет сократить время на этот процесс за счет автоматического отбора только тех записей, которые действительно требуется обновить, а также автоматизации других операций, например, по замене одних данных другими, выполнения арифметических действий и т.д.

Разработанный запрос представляет собой программный файл, который записывается в каталог. Имя этого файла высвечивается в окне запросов на панели УЦ. Запрос на отбор данных можно использовать многократно, посредством выполнения созданного и записанного на диске программного файла запроса.

Запросы можно создавать с помощью шаблона файла (запросы по образцу) и в специальном окне, которое размещается справа в средней части экрана. Для каждого запроса на отбор данных нужно обязательно сформировать один **шаблон вывода** данных, в котором помещаются поля, выводимые на экран. В одном запросе часть критерия отбора данных можно помещать в шаблон файла, а остальную часть в окне. Запросы на обновление данных в отличие от запроса на отбор позволяют обрабатывать только один файл БД.

Для создания запроса необходимо выбрать позицию **<create>** в окне **Queries** панели УЦ. Предварительно целесообразно открыть файл, в котором требуется отбирать данные. На экране появится меню этого окна с позициями **Layout** (оформление запроса), **Fields** (поля) и др. Если Вы не открыли файл, то это можно сделать, выбрав позицию **Add file to query** в меню **Layout** (вспомните, что значит выбрать позицию меню). На экране высветится список имен файлов в текущем директории. Выбор имени этого файла в окне приводит к его открытию и выводу его шаблона на экран. Таким же способом выводятся шаблоны и других файлов. Присутствие на экране шаблонов нескольких файлов требуется, например, при их связке. Шаблон файла можно удалить с экрана, выбрав позицию **Remove file from query** этого же меню.

Ввод поля файла в шаблон вывода выполняется клавишей **F5** после предварительной установки курсора в требуемое поле в шаблоне файла. Удаление поля из шаблона вывода также выполняется клавишей **F5** после предварительной установки курсора в соответствующее поле шаблона вывода. Ввод и удаление поля из шаблона вывода можно также выполнить с помощью позиций **Add fields to view** и **Remove field from view** меню **Layout**, предварительно установив курсор в соответствующее поле в шаблонах файла или вывода. Сразу все поля файла можно поместить в шаблон вывода клавишей **F5**, предварительно установив курсор в шаблоне файла под его именем. Повторное нажатие этой клавиши в положении курсора под именем файла вызывает удаление всех полей из шаблона вывода.

Для перемещения одного поля в другое место в шаблоне вывода нужно установить на него курсор, нажать клавишу **F6** (высветить), затем нажать клавишу **F7**, переместить курсор на новое место и нажать **Enter**. Перемещение нескольких полей выполняется аналогично после их подсветки, которая осуществляется передвижением по полям навигационными клавишами после

нажатия клавиши **F6**. Подсветку также нужно зафиксировать клавишей **Enter**.

Если все шаблоны файлов не умещаются на одном экране (их может быть до 8), то переход к следующему экрану выполняется клавишей **PgDn**, а к предыдущему - **PgUp**.

Движение по полям шаблонов выполняется, как и по полям файла БД клавишами **Tab** (вправо) и **Shift-Tab** (влево), в крайнее левое и правое поля - клавишами **Home** и **End**, а переход из шаблона файла в шаблон вывода и обратно - клавишами **F3** и **F4**.

Формирование шаблона вывода целесообразно выполнять в два этапа: сначала вводить сразу все поля файла и затем удалять ненужные. Как уже упоминалось, ввод всех полей осуществляется клавишей **F5** после предварительной установки курсора в шаблоне файла под его именем. Повторное нажатие клавиши **F5** в том же положении курсора вызывает удаление всех полей из шаблона вывода. Клавиша **F5** позволяет также вводить и удалять поля файла по одному в шаблон вывода после предварительной установки курсора в требуемое поле в шаблоне файла.

Кроме обычных полей в шаблон вывода можно помещать вычисляемые поля. В эти поля заносятся данные, получаемые в результате выполнения различных (не только арифметических) операций. Вычисляемые поля создаются в специальном шаблоне. Для этого следует выбрать позицию **Create calculated fields** меню **Fields**.

В пустую графу шаблона вычисляемых полей вводится с клавиатуры вычисляемое выражение, которое включает одно или несколько имен полей файла БД, и операторы выполнения действий с ними. В число этих операторов входят арифметические и логические операторы, а также функции dBASE IV. Имена полей, операторы и функции можно также ввести в шаблон из таблицы, высвечиваемой клавишами **Shift-F1**. Передвижение по этой таблице осуществляется навигационными клавишами, а ввод значения - клавишей **Enter**. Следует отметить, что с помощью этой таблицы чаще всего можно ввести не целое выражение, а только его элементы, которые обычно нужно редактировать для формирования синтаксически правильного вычисляемого выражения.

После сформирования выражения вычисляемого поля ему требуется присвоить имя, что выполняется выбором позиции **Edit field name** меню **Fields** (высвечивается окно, в которое вводится имя поля). Имена полей должны быть уникальными. Для перемещения созданного вычисляемого поля в шаблон вывода используются те же клавиши и меню, что и для обычных полей.

Как уже упоминалось, условия (критерии) для отбора данных из файлов БД в запросе можно сформировать двумя способами: с помощью шаблона файлов (так называемый запрос по образцу) и в специальном окне - или их комбинацией.

Возможности по формированию условий для отбора этими способами различны. В шаблоне файла можно, кроме формирования условия отбора, еще задать выполнение некоторых дополнительных действий (осуществлять связку нескольких файлов, выполнять операции по агрегированию данных и др.). Этим способом легче составлять запрос, так как не требуется вводить логические операторы. Однако только в специальном окне можно включать в поисковое выражение поля типа **Memo**, но нельзя задавать никаких действий помимо формирования условия и труднее формировать сложное условие, так как требуются некоторые знания математической логики (логические операторы и способы построения сложных запросов).

Сложный запрос состоит из элементарных условий, которые объединяются логическими операторами. В общем случае элементарное условие включает имя поля, следующий за именем оператор и поисковое значение поля.

При формировании запроса с помощью шаблона файлов элементарное условие вводится в графу соответствующего поля без имени поля. Размещение элементарных условий в одной строке шаблона соответствует логической связке **.AND.** **(И)**, а в разных строках шаблона - связке **.OR.** **(ИЛИ)**. Следующая строка шаблона файла раскрывается навигационной клавишей **Вниз**. Операторы элементарных условий приведены в табл. 2.11.

Оператор "=" в элементарной условии, помещаемом в шаблон файла, можно не ставить, так как предполагается, что если введено значение без оператора, то такое условие означает отбор записей с данными, равными этому значению. После оператора **Like** (подобно) записывается значение - строка в кавычках, которую требуется найти. Эта строка может включать знаки замещения * и ? при поиске символов. Знак * замещает любое количество символов, а знак ? - один символ.

Таблица 2.11. **Операторы элементарных условий**

Оператор	Описание
>	Больше чем
<	Меньше чем
=	Равно
<> или #	Не равно
>=	Больше или равно
<=	Меньше или равно
$	Содержит
Like	Подобно
Sounds like	Звукоподобно

С оператором **Sounds like** также вводится символьная строка в кавычках. Этот оператор выполняет сравнение заданной и искомой строк в соответствии с кодами, определяющими звучание этих строк. Если коды звучания двух разных строк одинаковые, то эти строки звукоподобны и условие поиска выполняется.

При формировании элементарного условия вместо явного значения можно задавать его неявно в виде имени поля. Такой тип условия требуется, например, при отборе записи со значениями некоторого числового поля, превышающими все значения другого числового поля.

Примером служит запись элементарного условия в поле **pole1** шаблона файла **>=pole2 - 5** для отбора записей со значениями в поле **pole1**, больших или равных любым значениям в поле **pole2** минус пять. Вместо имени поля можно ввести имя любой переменной, например, **zz**, и это же имя поместить в графу соответствующего поля шаблона файла. Если в элементарное условие входит поле из другого файла, то перед его именем необходимо поставить имя файла со стрелкой вправо по форме: **<имя файла>-><имя поля>**.

При формировании запроса в специальном окне применяются обычные правила составления логического условия. Сложное условие представляет собой комбинацию элементарных условий, соединенных логическими связками **.AND.**, **.OR.** и **.NOT.** Элементарное условие в отличие от условия, вводимого в шаблон файла, записывается по полной форме, т.е. из трех компонентов: имени поля, оператора и значения. При формировании сложного условия могут использоваться круглые скобки для разграничения действия логических операторов.

Обычно при выполнении запроса осуществляется отбор (фильтрация) записей файла БД в соответствии с заданным условием. Можно запрос на фильтрацию превратить в запрос на поиск первой записи, удовлетворяющей заданному условию, если после критерия отбора (элементарного условия) поместить слово **FIND**.

Специальное окно для формирования условия отбора записей вызывается выбором позиции **Add condition box** меню **Condition**. Позиция **Delete condition box** этого же меню позволяет удалить окно. Установка клавишей **Enter** значения **YES** позиции данного меню **Show condition box** открывает окно, а значения **NO** его закрывает. После закрытия окно удаляется с экрана, но не из памяти, а на экране остается маркер - слово **CONDITION**. Впоследствии окно можно активизировать, переместив курсор на маркер клавишами **F3** или **F4** или установив значение **YES** в позиции **Show condition box**.

Как уже упоминалось, при формированиии запроса с помощью шаблона файла можно, кроме элементарных условий, задавать еще и выполнение дополнительных действий: вывод

отобранных записей в упорядоченном виде, группирование записей, агрегирование сгруппированных числовых данных с помощью специальных функций, связку файлов, представленных в запросе своими шаблонами, по общему полю.

Задание вывода отобранных данных в упорядоченном виде выполняется вводом в одну или несколько граф шаблона файла операторов сортировки: **ascN** (в возрастающем порядке), **dscN** (в убывающем порядке), **ascdictN** (в возрастающем порядке без различия прописных и строчных букв), **dscdictN** (в убывающем порядке без различия прописных и строчных букв). Эти операторы можно набрать с клавиатуры или выбрать из таблицы, высвечиваемой после выбора позиции **Sort on this field** меню **Fields**. Если в графе шаблона файла уже есть элементарное условие для отбора данных, то оператор упорядочивания и условие должны разделяться запятой.

Группирование данных файла БД представляет собой подбор записей с одинаковыми значениями полей в один блок. По умолчанию сгруппированные блоки данных располагаются по возрастанию чисел для числовых полей или ASCII-кодов для символьных полей. Этот порядок можно изменить на обратный, введя альтернативный оператор упорядочивания. Для выполнения группирования выбирается позиция **Group by** после предварительной установки курсора в соответствующую графу шаблона файла. Отметим, что вместо выбора позиции **Group by** можно ввести эти слова в графу шаблона непосредственно с клавиатуры.

Операторы агрегирования включают: **SUM** - суммирование значений полей; **AVG**, **MIN**, **MAX** - определение среднего, минимального и максимального значений полей; **CNT** - подсчет количества отобранных записей. Часто операторы агрегирования используются в сочетании с оператором группирования. При этом группирование применяется по текстовым полям, а агрегирование по числовым полям этих же записей, что позволяет, например, вычислять различные функции агрегирования в группах (подытоги, минимальное, максимальное и среднее значения).

В нашем примере информационной задачи учета автоперевозок технические характеристики автомобилей и данные об их парке хранятся в отдельных файлах. При получении сведений о грузообороте можно использовать связь этих файлов по полю **MAR** (марка автомобиля) и из файла технических характеристик извлекать численные значения грузоподъемности автомобиля для расчета итоговых показателей за определенный период времени.

Связывание файлов по общему полю осуществляется вводом имени одной и той же переменной в связываемые поля файлов. Это можно выполнить с клавиатуры или выбором позиции **Create link by pointing** меню **Layout**. В последнем случае в качестве такой переменной автоматически вводится имя **LinkN**, где **N** - порядковый номер связи между полями файлов БД. После установления связи между файлами из них выводятся только те

записи, которые имеют одинаковое значение связанных полей. Если требуется вывести все записи связанных файлов, то перед одной из связывающих переменных надо поставить слово **Every**. В том случае, когда необходимо вывести только одну (первую) запись из группы записей с одинаковыми значениями связанных полей, следует поставить слово **First**.

Связывание файлов можно использовать как формальный прием при формированиии специальных типов запросов на отбор данных из одного файла, в котором требуется сравнить каждую запись со всеми другими в файле. В этом случае шаблон файла нужно дважды вызвать на экран. Дубликат помечается буквой **B** в верхнем левом углу шаблона файла. После связки этих двух шаблонов файлов по какому-либо полю формируется запрос, при выполнении которого производится сравнение каждой записи файла со всеми другими. В шаблон вывода такого запроса можно вводить одно и то же поле несколько раз. При этом высвечивается подсказка изменить имя поля. После изменения поле вводится в шаблон вывода, где указывается новое и старое имя.

Кроме полей в состав шаблона файла можно включить созданные ранее пользователем сложные индексы (простые индексы полей входят в состав списка указателей индексов в связанном множественном индексном файле и подключаются автоматически при выполнении соответствующих операций). Эти индексы представляют собой псевдополя, которые можно использовать для связи с другими файлами БД и сортировки. Для включения сложного индекса выбирается позиция **Include indexes** меню **Fields**. Созданные ранее сложные индексы высвечиваются в конце шаблона файла. В них стоит знак # и выражение индекса. Одновременно при выборе этой позиции знак # помещается в каждое индексированное поле (с простым индексом) для напоминания пользователю об индексации используемого файла БД.

Запись запроса в каталог на диске осуществляется выбором позиции **Save this query** меню **Layout** или позиции **Save changes and exit** меню **Exit**. Те же действия выполняются и при нажатии клавиш **Ctrl-End**. Перед записью высвечивается окно, в которое нужно ввести с клавиатуры имя запроса и нажать **Enter**. При выборе позиции **Abandon changes and exit** меню **Exit** осуществляется выход из экрана создания/модификации запроса без сохранения.

Для проверки правильности формирования условий отбора данных, введенных в запрос, можно, не выходя из экрана создания/модификации запросов, просмотреть отобранные записи, нажав клавишу **F2**. Возврат на экран создания/модификации запроса выполняется нажатием клавиш **Shift-F2** или выбором

позиции **Transfer to Query Design** меню **Exit** экранной формы **Browse** или **Edit**.

Активизация запроса осуществляется выбором его имени в окне **Queries** панели УЦ. После этого высвечивается горизонтальное меню с тремя позициями **Use view**, **Modify query**, **Display data**. Выберите позицию **Use view**. Для просмотра отобранных данных переместите курсор на позицию **Display data** и нажмите клавишу **Enter**.

Отобранные в результате выполнения запроса записи из нескольких файлов можно сохранить в виде одного файла БД с помощью позиции **Write view as database file** позиции **Layout**. Перед сохранением требуется ввести с клавиатуры имя файла. Для удобства пользователя можно ввести словесное описание созданного запроса, которое высвечивается при его запуске из окна **Queries** панели УЦ. Окно для ввода этого описания высвечивается выбором позиции **Edit description of query** меню **Layout**. Сохранение описания запроса после его ввода с клавиатуры выполняется клавишей **Enter**.

Как уже указывалось, можно формировать запросы не только на отбор записей, но и на обновление данных. В этих запросах нельзя изменять данные, полученные операторами агрегирования, и вычисляемые поля. Можно обновлять данные, полученные при сортировке по одному полю, если это поле проиндексировано.

При формировании запросов на обновление используются только шаблоны файлов. В число операций по обновлению данных входят: замена данных в полях записей, добавление записей из другого файла, пометка записей для их последующего удаления и уничтожение пометок на удаление. Для этого используются следующие операторы обновления: **REPLACE** (замещение данных), **APPEND** (добавление записей), **MARK** (пометка записей для удаления), **UNMARK** (уничтожение пометок для удаления записей). Операторы обновления следует вводить с клавиатуры в шаблон файла под именем файла БД или посредством выбора из списка, высвечиваемого выбором позиции **Specify update operation** меню **Update**.

Над именем файла в шаблоне после ввода оператора обновления появляется слово **Target**, означающее, что данный файл установлен в режим обновления в отличие от других файлов, которые могут оставаться на экране. При этом шаблон вывода, если он до этого был, исчезает с экрана. Для выполнения операции замещения данных необходимо в замещаемое поле ввести ключевое слово **with**, за которым следует выражение, определяющее вводимое значение. Кроме того, можно указывать фильтрующие условия, как и в запросе на отбор. В состав выражения могут входить числовые значения, имена полей данного файла и их комбинации с операторами и функциями, применимыми для соответствующих полей.

Например, можно заменить значение числового поля тем же значением, умноженным (разделенным, вычтенным и т.д.) на заданное число, или заменить значениями других полей, связанных арифметическими операторами. Содержимое символьных полей можно, например, замещать сцепленными или обработанными соответствующими функциями данными из других полей этого файла и т.д.

Операция добавления записей в файл с помощью запроса на обновление имеет смысл только в случае добавления записей из другого (передающего) файла, поскольку добавление данных в файл с клавиатуры можно осуществить после выполнения запроса на отбор данных (но только одного файла БД) или с помощью меню **Data** УЦ. Для этого шаблон передающего файла также должен быть на экране. В графы шаблона принимающего файла, в которые вводятся данные, надо ввести имена полей из передающего файла. Допускается также вместо имени поля вводить любое имя, однако в этом случае необходимо поместить это же имя в соответствующую графу поля передающего файла.

Если в одно поле принимающего файла требуется поместить данные из нескольких полей передающего файла, то в графе этого поля в шаблоне принимающего файла записывается выражение сцепления полей в виде имен полей, разделенных знаком "+". Вообще в графах шаблона принимающего файла можно записывать любые выражения (включающие операторы и функции), допустимые для действий с этими типами полей. В графах шаблона передающего файла можно записывать условия отбора передаваемых записей.

При выполнении операции пометки записей, предназначенных для удаления, можно вводить условия отбора этих записей. Если эти условия не введены, то помечаются все записи файла. Удаление помеченных записей из файла БД можно выполнить, выбрав позицию **Erase marked records** подменю **Organize** из меню создания структуры файла БД. Более короткий способ вызова этого подменю - клавишами **Shift-F2** из окна **Data** после установки курсора на имя файла БД.

Уничтожение пометок на удаление записей выполняется в соответствии с заданными в запросе условиями отбора записей. Если условия не заданы, то уничтожаются все пометки на удаление.

Выполнение запроса на обновление осуществляется выбором позиции **Perform the update** меню **Update**. Предварительно выводится предупреждающее сообщение и после подтверждения запрос выполняется. Можно записать обновленный файл БД на диск, выбрав позицию **Write view as database file** меню **Layout**.

Сохранение запроса на обновление в каталоге, выход без сохранения, выполнение запроса без сохранения и ввод словесного описания выполняются так же, как и для запроса на отбор

данных. После сохранения в каталоге УЦ запрос на обновление для отличия от запросов на отбор помечается знаком * слева от имени.

Ранее созданный запрос можно выполнить с панели УЦ, выбрав его имя в окне запросов. После этого высвечивается меню с позициями **Modify queries** (для перехода к модификации запроса) и **Display data** (на выполнение запроса с выводом данных на экран).

Более короткий способ перехода к выполнению (модификации запроса) - клавишами **F2** (**Shift-F2**) после предварительной установки курсора на имя файла запроса в окне на панели УЦ.

Для усвоения материала создайте запросы на отбор данных на водителей из файла **MBD1** по следующим признакам:

стаж работы больше 7 лет (поле S_R);

категория водительских прав В и С (поле KAT);

категория водительских прав Е или А;

стаж работы больше 5 лет и категория Е.

На учебной дискете эти запросы записаны под именами **Q01-Q04**. Если Вы используете для записи своих решений эту дискету или директорий **PRIMER** на жестком диске, в который записаны примеры учебной дискеты, то надо давать другие имена, чтобы учебные файлы не были заменены новыми файлами. В дальнейшем условимся, что файлам запросов, как и файлам БД, будут даваться имена, под которыми они записаны на учебной дискете, но с добавлением впереди буквы **M**, например файлам **Q01-Q04** будут присвоены имена **MQ01-MQ04**.

Для формирования первого запроса в поле **S_R** шаблона файла введите условие >7. Теперь остается только присвоить имя запросу **MQ01** и сохранить в памяти, выбрав позицию **Save changes and exit** меню **Exit**. Ввод имени выполняется с клавиатуры в окно **Save as**: по следующей форме **D:\DBASEIV\PRIMER\MQ01**. В этом окне имя диска и текущий маршрут по директориям высвечиваются по умолчанию, поэтому остается ввести только имя запроса.

```
Layout   Fields   Condition   Update   Exit            12:56:37 pm
┌──────────┬─────────────┬───────────────┬─────────────┐
│Mbd1.dbf  │ D_R         │ S_R           │ KAT         │
└──────────<─────────────┼───────────────┼────────────>
           │             │>7             │             │
```

При формировании элементарных условий с использованием категории водительских прав нужно применять оператор отношения **$ (содержит)**, поскольку буквы нескольких категорий, которые имеет водитель, записываются в одно поле. Во втором запросе используйте этот оператор: в графу **KAT** шаблона файла поместите логические условия **$"B",$"C"** (буквы латинские).

```
 Layout   Fields   Condition   Update   Exit          12:57:31 pm
┌─────────┬──────────────────────┬──────────────────────┬────────────────────┐
│Mbd1.dbf │ KAT                  │ AUTO                 │ N_Z                │
└─────────<──────────────────────┼──────────────────────┼────────────────────┤
          │$"B",$"C"             │                      │                    │
```

Третий запрос отличается от второго типом логической связки .OR. (ИЛИ), которая реализуется размещением элементарных условий **$"E"** и **$"A"** в разные строки шаблона. Надо ввести одно из них (безразлично какое) в первую строку шаблона под именем **KAT**, затем нажать клавишу **Вниз**. Откроется вторая строка, в которую надо поместить второе элементарное условие.

```
 Layout   Fields   Condition   Update   Exit          12:58:29 pm
┌──────────────┬──────────────────┬──────────────────────┬───────────────────┐
│Mbd1.dbf      │ D_R              │ S_R                  │ KAT               │
└──────────────<──────────────────┼──────────────────────┼───────────────────>
               │                  │                      │$"E"               │
               │                  │                      │$"A"               │
```

В четвертом запросе введите условие **>5** в графу **S_R**, а условие **$"E"** в графу **KAT** (связка условий по **"И"**).

```
 Layout   Fields   Condition   Update   Exit          12:59:54 pm
┌──────────────┬──────────────────┬──────────────────────┬───────────────────┐
│Mbd1.dbf      │ D_R              │ S_R                  │ KAT               │
└──────────────<──────────────────┼──────────────────────┼───────────────────>
               │                  │>5                    │$"E"               │
```

```
┌View─────┬────────────┬────────────┬────────────┬────────────┐
│MQ04     │Mbd1->      │Mbd1->      │Mbd1->      │Mbd1->      │
│         │NOM         │FAM         │NAM         │OTC         │
└─────────┴────────────┴────────────┴────────────┴────────────>
```

В качестве дополнительных примеров на поиск данных создайте запросы с включением поля даты и поля типа **Memo**. Например, отберем водителей по признакам:

старше 45 лет;

старше 25 и моложе 45 лет;

имеющих категорию водительских прав B и моложе 35 лет:

имеющих категорию водительских прав E или моложе 30 лет;

родившихся в Москве;

родившихся в Москве, имеющих категорию водительских прав A и старше 40 лет.

Запросы с датой можно сформировать двумя способами: с указанием даты рождения или составив выражение, в которое входит возраст и функция, возвращающая текущую дату С помощью первого способа сформируйте элементарное условие вида: **<=** **{01/01/45}** и поместите его в графу **D_R** шаблона файла **MBD1**. Присвойте этому запросу имя **MQ05**. Следует иметь

в виду, что данные типа даты при формировании поискового выражения должны заключаться в фигурные скобки, что обозначает именно этот тип данных в отличие от символьных и числовых.

Во втором способе сформируйте выражение **< = DATE() - 45*365**, которое также поместите в графу **D_R**. Это выражение означает вычисление даты, являющейся разностью между текущей датой и возрастом, выраженным в днях. В этом случае автоматически будет определяться дата рождения, соответствующая заданному возрасту. Запишите этот запрос под именем **MQ06**.

Запросы на диапазон возраста являются сложными и формируются из двух элементарных условий, объединяемых логической связкой **.AND.** (**"И"**). Напомним, что в шаблоне файла эта логическая связка указывается неявно, в данном случае (для второго из дополнительных запросов) размещением обоих условий в одной графе через запятую: **> = {01/01/45}, < = {01/01/65}**. Запишите этот запрос под именем **MQ07**.

Как и в предыдущем запросе можно использовать функцию текущей даты. Тогда поисковое выражение в этом запросе, которому дадим имя **MQ08**, будет выглядеть так:

> = DATE() - 45*365,< = DATE() - 25*365

При решении третьего примера (запишем его под именем **MQ09**) сформируйте сложное условие типа "И", состоящее из двух элементарных, относящихся к разным полям, которые размещаются в одной строке в соответствующих полях: в поле **KAT $ "B"**, а в поле **D_R > = {01/01/55}**.

Четвертый пример отличается от третьего по своей структуре тем, что в нем нужно сформировать условие типа ИЛИ. В этом запросе, которому дайте имя **MQ10** элементарные условия (для поля **KAT $ "E"**, а для поля **D_R > {01/01/60}**) размещаются в разных строках шаблона.

В следующем примере требуется осуществить поиск по полю типа **Memo**. Поиск по этому полю можно сформировать только в специальном окне. Для этого нужно сначала активизировать меню клавишей **F10**, затем выбрать позицию **Add condition box** меню **Condition**. В высветившееся окно с клавиатуры введите элементарное логическое условие полной структуры: **"Москве" $ AUTO**. Заметим, что для поля типа **Memo** в элементарном условии значение ставится на первом месте. Для демонстрации этого примера в поле **AUTO** должны быть введены данные о месте рождения. Запишите этот запрос под именем **MQ11**.

В шестом примере требуется сформировать сложное условие, состоящее из трех элементарных, объединенных логическими связками **.AND.**:

"Москве" $ AUTO .AND. "A" $ KAT .AND. D_R < = {01/01/50}

Запишите это условие в окне и сохраните под именем **MQ12**. Этот же запрос можно сформировать как комбинацию части условия с полем типа **Memo** в окне, а остальной части в шаблоне файла. Сохраните этот запрос под именем **MQ13**.

Как уже упоминалось, с помощью шаблона файла можно связать несколько файлов и создать запрос на отбор данных из связанных файлов. Проиллюстрируем это на примере двух файлов **MBD1** и **MBD2**. Для этого свяжите их по общему полю **N _ Z** и создайте запрос на отбор данных по тем автомобилям, которые не исправны (поиск по полю **"Состояние" - SOS** со значением **"Не исправен"**).

Сначала выведите на экран (в любой последовательности) шаблоны файлов **MBD1** и **MBD2**. Напомним, что вывод шаблона файла осуществляется позицией **Add file to query** в меню **Layout**. Не забудьте также ввести в шаблон вывода связываемые поля. Если в этот шаблон не войдут какие-либо из полей связанных файлов, то они не будут выводиться не только на экран, но и вообще не будут присутствовать в выборке. Таким образом, фактически в этом случае связка не будет реализована. Формирование полей шаблона вывода выполняется последовательно для каждого файла. При этом целесообразно сначала ввести все поля файла, а затем удалить ненужные. Поля каждого следующего файла будут добавляться в шаблон вывода.

Можно также, наоборот, вводить в шаблон вывода только те поля, которые будут нужны. Напомним, что для ввода поля в шаблон вывода нужно клавишами **F3/F4** переместить курсор на шаблон файла, затем клавишами **Tab** или **Shift-Tab** передвинуть курсор на нужное поле и нажать клавишу **F5**. Для перенесения поля, на котором установлен курсор, можно вместо клавиши **F5** использовать позицию **Add field to view** меню **Fields**. Для нашего примера в шаблоне вывода обязательно должны присутствовать поля, по которым осуществляется связка и поиск **N _ Z**, **SOS**, **MAR** и **G _ N**, а также другие необходимые для вывода поля.

Для связки файлов **MBD1** и **MBD2** по общему полю **N _ Z** с помощью клавиш **Shift-Tab** надо переместить курсор на поле **N _ Z**, выбрать позицию **Create link by pointing** меню **Layout** (в графе поля высветится слово **LINK1**), клавишей **F3** перейти на шаблон файла **MBD1**, установив курсор в графу поля **N _ Z** и нажав клавишу **Enter** (в графе также высветится **LINK1**). Связка файлов выполнена.

При формировании условия на отбор записей нужно перейти клавишей **F4** на шаблон файла **MBD2** и в поле **SOS** ввести условие отбора записей: **"Не исправен"** (кавычки вводить надо, так как это поле символьного типа).

После сформирования запроса можно ввести его описание, например: **"Связка файлов MBD1, MBD2 по общему полю N _ Z и отбор данных о неисправных автомобилях"**. Для этого сначала

нужно выбрать позицию **Edit description of query** меню **Layout** . Теперь остается только присвоить имя запросу, например **MQ14**, и сохранить в памяти, выбрав позицию **Save changes and exit** меню **Exit**. Ввод имени выполняется с клавиатуры в высвеченное окно **Save as** по форме: имя диска:\маршрут по директориям\имя запроса, например **D:\DBASEIV\PRIMER\MQ14**.

Попробуйте теперь самостоятельно создать запрос на связку файлов БД: **MBD1**, **MBD2** и **MBD3** по общему полю **N_Z**; **MBD2** и **MBD4** по полю **MAR** - и отобрать данные о перевозках, выполненных водителями на расстояние свыше 300 км. Сохраните этот запрос под именем **MQ15**.

При создании связки нужно иметь в виду, что с помощью меню связка файлов осуществляется парами и в общем поле каждой следующей пары автоматически записывается слово **LINKn**, где **n** - порядковый номер связываемой пары. Таким образом, в данном запросе в поле **N_Z** файла **MBD1** будет стоять слово **LINK1**, в этом же поле файла **MBD2** два слова **LINK1** и **LINK2**, разделенные запятой, а в файле **MBD3** - слово **LINK2**. В общем поле **MAR** файлов **MBD2** и **MBD4** будет стоять слово **LINK3**.

Помимо меню связку файлов можно выполнить вручную с клавиатуры. Для этого нужно в общее поле связываемых файлов ввести какое-либо одно имя, например, в поле **N_Z** - имя **L1**, а в поле **MAR** - имя **L2**. Ниже, в разделе разработки отчетов будет описано создание запросов двумя этими способами.

Следует отметить, что существенным недостатком запросов на отбор данных из связанных файлов, создаваемых с помощью УЦ, является то, что они не дают возможности пользователю редактировать отобранную информацию.

Опытные пользователи, знающие язык dBASE IV, могут расширить возможный диапазон действий, выполняемых запросами посредством модификации созданных и сохраненных программных файлов запросов (имеют расширение **.qbe**) с помощью текстового редактора.

2.6. ЭКРАННЫЕ ФОРМЫ ВВОДА И ВЫВОДА ДАННЫХ

В предыдущих разделах ввод информации в файлы БД и вывод ее на экран выполнялся в виде стандартных экранных форм **Browse** и **Edit**. В dBASE IV имеется возможность разрабатывать собственные формы ввода и вывода данных на экран. В этих формах поля файлов БД можно размещать в любом месте экрана, сопровождая их пояснительными надписями, обводить в рамки и вводить другие оформительские элементы. Важной особенностью пакета dBASE IV является возможность вывода в окна на экране и редактирование полей типа **Memo**.

Кроме того, при установке курсора в поле файла БД можно высвечивать сообщения внизу экрана об особенностях и правилах ввода информации, контролировать ввод информации и т.д. Каждая экранная форма предназначается для ввода или вывода данных из определенного файла БД, имя которого записано в этой экранной форме. Этот файл можно изменить, модифицируя экранную форму.

Разработанные формы представляют собой программные файлы, которые можно записать на диск и затем использовать многократно для просмотра и обновления информации на экране в удобном для пользователя виде.

Перед созданием экранной формы целесообразно активизировать (открыть) файл БД или выборку, которые будут выводиться на экран в разрабатываемой форме. Однако это не обязательно. Можно открыть требуемый файл и из меню разработки экранной формы. Разработка экранной формы начинается с выбора позиции **<create>** окна **Forms** панели УЦ. После этого высвечивается горизонтальное меню, включающее позиции **Layout**, **Fields**, **Words**, **GoTo** и **Exit** и экран, покрытый мелкой сеткой. Выбор этих позиций приводит к открытию различных меню, позволяющих выполнять определенный круг действий по созданию экранной формы.

Мелкая сетка играет определенную роль при размещении различных элементов в экранной форме. Наличие сетки означает, что экран ничем не заполнен. Удаление сетки означает заполнение экрана пробелами. Так, перемещение текста клавишей **Пробел** в режиме **Ins** возможно только по экрану, свободному от мелкой сетки.

Удаление мелкой сетки осуществляется клавишей **Пробел**, а восстановление - клавишей **Del** или **Backspace**. Если два поля разделены мелкой сеткой, то они при выводе отделяются интервалом, равным числу позиций мелкой сетки между ними. Если мелкой сетки между пробелами нет, то поля при выводе прижимаются друг к другу. Кроме того, мелкая сетка служит основой для закрашивания областей экрана. Места экрана, в которых сетка отсутствует, нельзя раскрасить в любой цвет. При высвечивании экранной формы они будут иметь цвет периферийной части экрана.

Меню **Layout** предназначено для общего оформления экранной формы, реализации короткого способа ввода всех полей на экран и назначения файла БД, если он не был открыт ранее.

Существует два способа ввода полей на экран из файла БД - по одному полю или сразу всех полей. Все поля файла БД в том порядке, как они расположены в БД, можно ввести, выбрав позицию **Quick layout** меню **Layout**. В этом случае надо обязательно предварительно открыть файл БД. При этом поля располагаются в столбик, каждое поле - на одной строке экрана.

Затем эти поля можно, поочередно передвигая, более удобно разместить на экране. Если в текущий момент нет активного файла или с данной формой нужно использовать другой файл, то его можно открыть, выбрав из окна, высвечиваемого после выбора позиции **Use different database file or view** меню **Layout**.

Передвижение и копирование одного или нескольких полей осуществляется соответственно клавишами **F7** и **F8**, изменение размеров - клавишами **Shift-F7**. Перед перемещением и копированием одного поля или нескольких полей их надо высветить клавишей **F6**. Добавление полей по одному осуществляется с помощью позиции **Add field** меню **Fields**. Перед добавлением поля нужно установить курсор на то место, в котором будет располагаться поле. После выбора этой позиции высвечивается список полей файла БД, а после выбора имени поля появляется шаблон для изменения заданных по умолчанию параметров поля. Сохранение параметров поля выполняется клавишами **Ctrl-End**.

Более короткий способ для выполнения операций по добавлению поля - нажать клавишу **F5**, также предварительно установив курсор на место, в котором должно располагаться поле.

Удаление ненужного поля с экрана выполняется с помощью позиции **Remove field**. После выбора этой позиции высвечивается список полей файла БД. Выбор имени приводит к удалению поля. Удалить поле можно также клавишей **Del**, предварительно установив курсор на поле.

Поля типа **Memo** в экранной форме можно представить в скрытом виде, который обозначается на экране словом (маркером) **memo** (**MEMO**) или в явном виде в окне заданных размеров в любом месте экрана. Маркер **memo**, записанный строчными (прописными) буквами, означает, что это поле пусто (заполнено). Скрытое поле переводится в оконную форму представления клавишами **Ctrl-Home** после установки курсора на маркер. Нажатие клавиши **F9** переводит окно в полноэкранную форму представления. Для выхода из поля типа **Memo** следует нажать клавиши **Ctrl-End**.

Существуют два способа создания оконной формы поля типа **Memo**, которые соответствуют двум способам ввода полей на экран. Если используется способ ввода всех полей на экран с помощью позиции **Quick layout** меню **Layout**, то нужно установить курсор на поле типа **Memo** и выбрать позицию **Modify field** меню **Fields**. После этого высветится окно с позициями для установки параметров, в котором нужно в позиции **Display as** клавишей **Пробел** выбрать значение **WINDOW** и нажать **Ctrl-End**. Далее высветится окно поля **Memo** стандартных размеров, которое можно привести к нужным размерам или переместить. Для этого сначала надо установить курсор на рамку окна, затем нажать клавиши **Shift-F7** или **F7**. Затем можно приводить окно к

нужным размерам или перемещать его навигационными клавишами **Вверх**, **Вниз**, **Вправо** и **Влево**.

Если используется способ ввода полей в экранную форму по одному, то надо выбрать позицию **Add field** меню **Fields**, затем в высветившемся окне выбрать поле **Memo** и в появившемся после выбора поля шаблоне установить в позиции **Display as** клавишей **Пробел** значение **WINDOW** и нажать **Ctrl-End**. После этого надо переместить курсор в левый верхний угол предполагаемого окна **Memo** поля и нажать **Enter**, затем переместить курсор в правый нижний угол окна и зафиксировать клавишей **Enter**. В результате высветится рамка окна.

Создание скрытой формы **Memo** поля при использовании способа ввода полей по одному выполняется сначала так же, как и оконной формы, но вместо значения **WINDOW** в позиции **Display as** выбирается **MARKER**. После высвечивания рамки окна нужно переместить курсор в позицию, в которой должен быть маркер, и нажать **Enter**. Окно исчезнет и появится маркер поля **Memo**. При использовании способа ввода сразу всех полей перед созданием скрытой формы поля **Memo** нужно сначала сформировать оконную форму, как это описано выше. Затем установить курсор на рамку этого поля, выбрать позицию **Modify field** меню **Fields** и затем клавишей **Пробел** установить значение **MARKER** в позиции **Display as** шаблона параметров поля, не забыв сохранить установленное значение клавишами **Ctrl-End**. После этого нужно переместить курсор в место, в котором должно быть поле **Memo** и нажать клавишу **Enter**. Окно исчезнет и появится маркер **Memo** поля.

Созданные на экране поля **Memo** можно, как и другие, передвигать, копировать, изменять в размерах (исключая скрытую форму) и удалять с помощью позиции **Remove field** и **Modify field** меню **Fields** или с клавиатуры теми же клавишами, что и для обычных полей. С помощью позиции **Border line** меню **Fields** можно изменить вид рамки окна **Memo** поля, установив одинарную (**Single**), двойную (**Double**) линию или линию рамки из заданных символов (**Using specified character**).

Позиция **Modify field** меню **Fields** позволяет изменить различные параметры поля, в том числе шаблон и функции поля (кроме **Memo** и логических) и т.д. Шаблоны и функции поля определяют форму ввода и вывода данных в это поле. Шаблон поля представляет собой набор знаков в количестве, равном выводимой на экране ширине поля. Каждый знак действует на ту позицию поля, в которой он находится. Примером действия знаков шаблона служит ввод только цифр (знак **9**) или только букв (знак **A**) и т.д.

Для числовых полей можно использовать следующие знаки шаблона:

9 - ввод и вывод цифр и знаков чисел;

- ввод и вывод цифр, пробелов и знаков чисел,

. - определение положения десятичной точки при вводе и выводе чисел;

, - разделение при выводе трех порядков чисел;

* - замена при выводе лидирующих нулей чисел знаком *****;

$** - замена при выводе лидирующих нулей чисел знаком **$.

Для символьных полей используются следующие знаки шаблона:

9 - ввод и вывод цифр;

- ввод и вывод цифр, пробелов и знаков чисел;

A - ввод и вывод букв;

N - ввод и вывод цифр и букв;

! - преобразование строчных букв в прописные.

Для логических полей используются знаки шаблона:

L - ввод и вывод логических значений **T**,**F**,**Y** и **N**;

X - ввод и вывод любого символа;

Y - ввод и вывод логических значений **Y** и **N**.

Другие знаки в составе шаблона будут вводиться как символы данных в соответствующую позицию поля, если не задана функция поля **R**.

Изменение шаблона поля, заданного по умолчанию, осуществляется выбором позиции **Template** подменю **Modify field** меню **Fields**. Знаки шаблона вводятся в высвечиваемое поле с клавиатуры.

Функции поля являются дополнительным средством обработки данных и позволяют кроме установленного шаблоном режима ввода и вывода информации выполнять аналогичные и дополнительные действия по обработке вводимых и выводимых данных. Функции поля в отличие от шаблона служат для различных преобразований целых полей. Совместное использование функций и шаблонов поля дает возможность комбинировать устанавливаемые ими действия.

Функция поля задается выбором соответствующей позиции в окне, высвечиваемом после выбора позиции **Picture functions**. В начальном состоянии все функции выключены (состояние **OFF**). Выбор функции приводит к ее включению (состояние **OFF** меняется на **ON**).

Для числовых полей используются следующие функции:

L - вывод лидирующих нулей чисел;

Z - вывод пробелов вместо нулевого значения числа;

$ - вывод числа со знаком валюты;

^ - вывод числа в записи с порядком (научная запись числа).

Для символьных полей применяются следующие функции:

A - ввод только букв;

! - преобразование строчных букв в прописные;

R - ввод любых символов как данных в поле;

S - прокрутка навигационными клавишами информации в поле экранной формы;

М - выбор клавишей **Пробел** значений поля из созданного списка.

На практике для символьных полей часто используются функции поля **S** и **М.** Первая применяется для установки возможности прокрутки информации в поле на экране, размер которого меньше, чем исходный размер поля в файле БД, а вторая - для создания списка альтернативных значений поля. После их выбора высвечивается окно: в первом случае для ввода ширины поля, внутри которого требуется прокручивать данные, а во втором - для ввода списка значений поля, которые впоследствии при использовании формы можно будет вызывать клавишей **Пробел**. Значения списка вводятся с клавиатуры и разделяются запятыми.

Шаблон и функции поля не применяются для полей **Memo** и полей типа даты, а для логических полей можно использовать только шаблоны.

Кроме шаблонов и функций для каждого поля можно задать ряд установок, которые позволяют сделать более удобной работу с данными в созданной экранной форме: установить защиту от обновления содержимого поля, задать условие, разрешающее обновлять данные в поле, дать указание о выводе внизу экрана сообщения при входе курсора в это поле, копировать в поле текущей записи значение из предыдущей записи, установить начальное значение данных в поле, обеспечить проверку на диапазон вводимых числовых данных, задать условие, когда в поле можно ввести новые данные, задать вывод внизу экрана сообщения, если условие для ввода в поле данных не выполняется.

Эти установки задаются выбором позиций меню, высвечиваемого после выбора опции **Edit options** меню **Fields**. После выбора позиции в зависимости от ее назначения требуется ввести информацию или установить альтернативное состояние (значение). Фиксация установленного состояния (значения) и введенной информации осуществляется клавишей **Enter**.

Разрешение (запрет) обновления поля устанавливается выбором клавишей **Enter** слова **Yes (No)** в позиции **Editing allowed**. Условие для обновления поля вводится в окно, открывающееся после выбора позиции **Permit edit if**. В качестве условия можно записать любое допустимое выражение для полей файла БД. Ввод текста, который будет высвечиваться внизу экрана при перемещении курсора в поле (например, для пояснения специфики обновления данных) осуществляется выбором позиции **Message**. Установка копирования в поле значения из предыдущей записи выполняется выбором слова **Yes** в позиции **Carry forward**. Начальное значение поля вводится после выбора позиции **Default value**. Контроль диапазона вводимых числовых данных можно осуществить, установив минимально и максимально допустимые значения чисел или данных типа даты

после выбора позиций **Smallest allowed value** и **Largest allowed value**. Условие для ввода нового значения в поле и сообщение, которое будет высвечиваться внизу экрана, можно ввести после выбора позиций **Accept value when** и **Unaccepted message**. Сохранение всех данных, относящихся к полю файла БД, выполняется клавишами **Ctrl-End**.

Кроме полей файла БД в экранную форму можно вводить вычисляемые поля. Меню для создания вычисляемого поля высвечивается после выбора позиции **<create>** под заголовком **CALCULATED** в таблице из двух колонок, появляющейся в результате выбора позиции **Add fields** меню **Fields**. В этом меню после выбора позиции **Name** надо задать имя вычисляемого поля, после выбора позиции **Description** можно ввести с клавиатуры словесное описание вычисляемого поля и обязательно после выбора позиции **Expression** надо ввести выражение, определяющее вычисляемое поле. В этом выражении можно использовать все допустимые операторы применительно к соответствующим полям файла БД. Нельзя в выражении указывать ранее созданные вычисляемые поля. Для вычисляемого поля можно установить шаблон и функции поля. Знаки шаблона вычисляемого поля не отличаются от знаков шаблона обычного числового поля. В состав функций, которые могут использоваться для вычисляемого поля, входят две ранее описанные функции - **!** и **R**, а также новые функции:

T - удаление ведущих и конечных пробелов;

J - выравнивание содержимого поля вправо;

I - центровка содержимого поля.

Кроме обычных полей в экранную форму можно включать переменные. Значения переменных вводятся с клавиатуры и могут использоваться для различных операций с данными из полей файла БД. Это существенно расширяет возможности по обработке информации с помощью экранной формы. Для ввода в экранную форму переменной служит позиция **Insert memory variable**. Тип переменной определяется выбором из альтернативных значений клавишей **Пробел**. Кроме типа переменной после выбора соответствующей позиции следует также ввести имя переменной (**Name**), шаблон поля переменной (**Template**), функции поля (**Picture function**) и опции редактирования поля переменной (**Edit options**). Эти данные вводятся аналогично обычным полям файла БД. Как и для обычных полей, после ввода параметров переменной их надо сохранить клавишами **Ctrl-End**. После этого поле переменной появляется в том месте, где расположен курсор. Поле переменной, как и другие поля, можно перемещать в любое место экрана по общим правилам перемещения полей.

При разработке экранной формы в нее кроме полей и текстового материала можно вводить оформительские элементы, например рамки, полоски, расцвечивать различные области экрана

и т.д. Один такой элемент уже рассматривался выше - рамки для полей типа **Memo**. В меню **Layout** имеются позиции для рисования рамок (**Box**) и линий (**Line**) на экране вокруг других полей или текстового материала. После выбора позиции **Box** высвечивается окно для выбора трех типов линий, как и для рамки поля **Memo**. Рисование рамки осуществляется аналогично формированию окна поля **Memo**. Более широкие возможности для рисования на экране обеспечивает позиция **Line**, позволяющая делать рисунки из отрезков горизонтальных и вертикальных линий. Эти линии можно прервать для записи текста, а затем вновь продолжить. Действия по рисованию линий выполняются аналогично формированию рамки, т.е. сначала нужно зафиксировать клавишей **Enter** начало линии, затем нарисовать линию, набрать текст и завершить рисование также нажатием клавиши **Enter**. Если что-то не получается с линиями, например, не сходятся концы и т.п., лучше всего прервать рисование клавишей **Enter**, затем удалить ненужные элементы клавишами **Del** или **Backspace** и затем начать рисование с прерванной точки.

В экранной форме можно размещать поля, оформительские элементы и текстовый материал, выходящие за границы стандартного экрана. Размер экранной формы ограничен только емкостью свободной оперативной памятью ПК. Для удобства работы с многостраничными экранными формами имеются меню **Words** и **Goto**. С помощью меню **Words** можно устанавливать различный цвет полей, рамок и других областей экрана, позиционировать текст, изменять разметку экранной формы в верхней части экрана или удалять ее, добавлять или удалять строку текста, вводить текстовый материал в экранную форму из других файлов и записывать текст экранной формы во внешний текстовой файл. Меню **Goto** позволяет быстро перемещаться по строкам экранной формы, осуществлять поиск и заменять слова.

Для расцвечивания областей экранной формы используется позиция **Display** меню **Words**. Перед ее выбором необходимо высветить клавишей **F6** и навигационными клавишами ту область экрана, которую требуется закрасить. Установка цвета области выполняется передвижением курсора по цветовой гамме палитры для фона (**Background**) и текста (**Foreground**).

Позиционирование текста выполняется выбором опции **Position** меню **Words**. После этого появляется меню с позициями: **Left** - высвеченный участок текста передвигается в левый край, **Right** - в правый край, **Center** - центрируется.

Удаление разметки экрана выполняется выбором позиции **Hide ruler**, а ее изменение - выбором позиции **Modify ruler**. Добавление строки в месте расположения курсора осуществляется позицией **Add line**, а удаление - позицией **Remove line** меню **Words**. Более подробно о позициях **Hide ruler** и **Modify ruler** будет сказано в разделе разработки форм отчетов.

Ввод (вывод) текста из внешнего файла (во внешний файл), начиная с текущей позиции курсора, выполняется выбором позиции **Write/read text file**. После этого появляется меню с позициями **Write selection to file** для записи во внешний файл и **Read text from file** для считывания в экранную форму из внешнего файла. В обоих случаях в высветившемся окне необходимо набрать с клавиатуры имя внешнего файла или выбрать его из директориев дисков, высвечиваемых клавишами **Shift-F1**.

В меню **Goto** позиция **Goto line number** служит для перехода на строку с заданным номером, который нужно ввести с клавиатуры. Для поиска слова (или части слова по включению) используются позиции **Forward search** (поиск в прямом направлении) и **Backward search** (поиск в обратном направлении). После выбора позиции с клавиатуры нужно ввести искомое слово (или часть слова - поиск по включению), после чего поиск инициализируется клавишей **Enter**. Замена слов в тексте осуществляется позицией **Replace**. Эта операция выполняется совместно с поиском. Сначала требуется ввести старое слово, затем новое слово. После того как найдено первое встреченное заменяемое слово, пользователю на выбор представляются варианты замены (для этого надо ввести с клавиатуры соответствующую букву): заменить одно слово - **R**, пропустить одно слово - **S**, заменить все слова в тексте - **A**, отменить замену - клавишей **Esc**. При поиске слов можно задать условие не различать (или различать) прописные и строчные буквы с помощью установки значений **OFF** (**ON**) в позицию **Match capitalization**.

После ввода всей информации по созданию экранной формы ее нужно сохранить на диске с помощью позиции **Save this form** для последующего повторного использования. Перед сохранением экранной формы ей нужно с клавиатуры присвоить имя в окне **Save as:** по форме имя диска:\маршрут по директориям\имя экранной формы, например, **D:\DBASEIV\PRIMER\MF1**.

Для идентификации созданной экранной формы можно ввести ее словесное описание с помощью позиции **Edit description of the form**. Это словесное описание будет высвечиваться при выборе формы в окне **Forms** панели УЦ. Сохранение экранной формы можно также выполнить перед выходом в УЦ с помощью позиции **Save changes and exit** меню **Exit**. В этом случае, если данная форма ранее уже была записана на диск, то окно с этим именем не высвечивается и осуществляется перезапись новой формы на старую с тем же именем. Позиция **Abandon changes and exit** служит для выхода в УЦ без сохранения разработанной формы.

Более короткий способ высвечивания созданной экранной формы без покидания экрана создания/модификации экранной формы реализуется нажатием клавиши **F2**. Возврат к экрану

создания/модификации формы выполняется клавишами **Shift-F2** или выбором позиции **Return to Form Design** меню **Exit**.

Запуск экранной формы можно также осуществить с панели УЦ. После выбора имени экранной формы высвечивается меню с позициями **Modify forms** для перехода к изменению экранной формы, **Display data** для высвечивания экранной формы.

Продемонстрируем теперь, как создать простую экранную форму, показанную на рис. 2.4. Откройте файл **MBD1**. Установите курсор на позицию **<create>** в окне **Forms** и нажмите клавишу **Enter**. Высветится экран создания/модификации экранной формы. После входа в экран создания/модификации экранной формы автоматически активизируется вертикальное меню, прикрепленное к позиции **Layout** горизонтального меню. Выберите позицию **Quick Layout** и нажмите **Enter**. Слева на экран в столбик выведутся все поля файла **MBD1**.

Теперь расположите поля на экране, например так, как показано на рис. 2.4. Перед перемещением любого поля (кроме поля типа **Memo**, размещение которого описано ниже) надо установить на него курсор, нажать клавишу **F6**, затем клавишу **F7**, переместить курсор в нужное место и нажать **Enter**. При перемещении полей в нижней части экрана может высвечиваться сообщение: **Delete covered text and fields (Y/N)** ?, означающее, что на новом месте уже что-то находится, например, часть этого же поля. Для подтверждения перемещения нажмите клавишу **Y**.

```
Layout   Fields   Words   Go To   Exit                 12:53:39 pm
Для выхода в меню нажмите клавишу F10

        Табельный  номер  9999

      Фамилия                Имя            Отчество        Дата рожд.
 ---------------------------------------------------------------------
| XXXXXXXXXXXXXXX  XXXXXXXXXXXXXXX  XXXXXXXXXXXXXXX     DD/MM/YY     |
 ---------------------------------------------------------------------
              Стаж работы водителем (лет)       - 99
              Категория водительских прав       - XXXXXXXXXXX
              Номерной знак автомобиля          - XX-XX XXX

                    А В Т О Б И О Г Р А Ф И Я
 ---------------------------------------------------------------------
|xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx|
|xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx|
|xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx|
|xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx|
|xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx|
 ---------------------------------------------------------------------

Form   |C:\dbase\MF1        |Row:0 Col:0     |File:Mbd1 |        Ins
       Add field:F5  Select:F6  Move:F7  Copy:F8  Size:Shift-F7
```

Рис. 2.4. Экранная форма данных о водителях.

Как уже упоминалось, поля типа **Memo** можно представить на экране в форме окна и в скрытом виде, когда высвечивается только маркер этого поля - слово **MEMO**. Опишем, как разместить поле **Memo AUTO** в оконной форме. Для этого установите на него курсор и выберите позицию **Modify fields** меню **Fields**. В высветившемся меню установите курсор на позицию **Display as** и клавишей **Пробел** выберите опцию **WINDOW**. Затем выберите позицию **Edit options** этого же меню и установите значение опции **Editing allowed - Yes** для разрешения редактирования этого поля **Memo** и нажмите **клавиши Ctrl-End** дважды. В результате появится окно **Memo** поля стандартных размеров. Для приведения окна к нужным размерам установите курсор на рамку окна, после чего окно подсветится. Далее нажмите клавиши **Shift-F7** и навигационными клавишами (**Вправо**, **Влево**, **Вверх**) установите требуемые размеры окна и нажмите клавишу **Enter**. Для перемещения окна в нужное место экрана нажмите клавишу **F7** и передвиньте окно навигационными клавишами, не забыв зафиксировать новое положение клавишей **Enter**.

Сделайте пояснительные надписи к полям. Для акцентирования полей можно их обвести рамкой. Для этого выберите позицию **Line** меню **Layout**, а в высветившемся при этом вертикальном меню - позицию **Single line**. Установите курсор на нужную позицию экрана и нажмите клавишу **Enter**. Начальная позиция линии зафиксирована. Затем навигационными клавишами нарисуйте рамку. После окончания рисования не забудьте зафиксировать ее клавишей **Enter**. Если при рисовании рамок и линий сразу не удастся сделать требуемый рисунок (например, не совпало начало линии с концом), то завершите рисование, удалите ненужные отрезки прямых клавишами **Del** или **Backspace** и начните рисование с прерванной точки. После завершения оформления полей на новом месте сотрите уже ненужные теперь имена этих полей **NOM**, **FAM**, **NAM**, **OTC**, **D_R**, оставшиеся на прежнем месте.

Для обеспечения более полной информации о содержании полей экранной формы можно предусмотреть высвечивание сообщения внизу экрана при перемещении курсора на поле. Например, для поля **NOM** введите сообщение: "**Служит идентификатором водителя**". Для этого сначала выберите позицию **Modify Field** меню **Fields**, после чего высветится список полей, из которого выберите поле **NOM**. Затем в высветившемся меню выберите позицию **Edit options**. После этого появится еще одно вертикальное меню, в котором выберите позицию **Message** и введите этот текст. Введенный текст надо зафиксировать клавишей **Enter** и затем сохранить клавишами **Ctrl-End**. Чтобы выйти из меню нажмите еще раз **Ctrl-End**, что означает сохранение введенных в этом меню данных.

Шаблоны полей играют важную роль при вводе данных. Они позволяют сократить количество вводимых вручную символов,

организовать контроль ввода и обеспечить наглядность вывода данных. Например, в шаблоне поля **NOM** задайте функцию вывода ведущих нулей, поскольку по умолчанию устанавливается шаблон обычного числа. Для этого переместите курсор на поле **NOM**, выберите позицию **Modify Field** меню **Fields**, в высветившемся меню - позицию **Picture functions**, а затем в следующем меню - опцию **Show leading zeroes**, в которой с помощью клавиши **Пробел** установите слово **ON**. Для завершения установки нажмите два раза клавиши **Ctrl-End**.

Аналогично для полей **FAM, NAM, OTC** установите функцию запрета на ввод чисел в поле. Для этого в высветившемся меню после выбора позиции **Picture functions** меню переведите позицию **Alphabetic characters only** в **ON**-состояние клавишей **Enter** и затем нажмите два раза клавиши **Ctrl-End**.

Контроль ввода данных можно установить с помощью запрета ввода определенных значений. Например, требуется запретить ввод данных на водителей старше 55 и моложе 20 лет. Для этого переместите курсор на поле **D_R** и нажмите **F5** в качестве короткого выхода на модификацию установок поля. Выберите позицию **Edit options**, а в высветившемся вертикальном меню позицию **Accept value when**, и введите условие

D_R >= DATE() - 55*365 .AND. D_R <= DATE() - 20*365

где функция **DATE()** возвращает текущую дату.

Чтобы при вводе в поле **D_R** даты, не удовлетворяющей заданному диапазону значений, высвечивалось сообщение, выберите опцию **Unaccepted message**, введите текст "**Не подходит по возрасту!**".

Для облегчения процесса ввода данных целесообразно в шаблон помещать повторяющиеся символы. Например, при вводе номерных знаков автомобиля в поле **N_Z** каждой записи можно ввести разделитель цифр номерного знака и пробел, отделяющий цифровую и символьную части номера. Для этого измените шаблон ввода-вывода данных поля **N_Z**. Вместо третьего символа шаблона "**X**" введите "-", а вместо шестого символа "**X**" - пробел и затем нажмите клавиши **Enter** и **Ctrl-End**. После выполненных изменений шаблон поля **N_Z** будет выглядеть так: **XX-XX XXX**. Теперь при вводе с клавиатуры номерного знака пользователю не нужно ставить разделитель цифр и пробел между буквами и цифрами, они будут вводиться автоматически.

Экран можно раскрасить в различные цвета. Для этого с помощью клавиши **F6** и навигационных клавиш высветите ту область экрана, которую необходимо раскрасить. Затем выберите позицию **Display** меню **Words** и из палитры цветов подберите требуемые цвета текста и фона. После выбора цветов не забудьте зафиксировать их клавишами **Ctrl-End**.

Создание экранной формы в основном закончено. Можно лишь порекомендовать в левой части верхней строки экранной

формы ввести текст сообщения о том, как вызвать меню экранной формы, чтобы выйти в УЦ, например: "Для выхода в меню нажмите клавишу F10".

Для идентификации экранной формы можно ввести ее описание, которое будет высвечиваться в нижней строке УЦ при установке курсора на имя формы. Для этого выберите позицию **Edit description of form** меню **Layout** и введите словесное описание формы. Сохранение файла экранной формы на диске можно выполнить выбором позиции **Save this form** меню **Layout** или позиции **Save changes and exit** меню **Exit**.

После выбора позиции для сохранения формы высвечивается окно **Save as**, в которое нужно ввести имя формы после имени диска и маршрута по директориям, например, **D:\DBASEIV\PRIMER\MF1**, и нажать клавишу **Enter**, после чего созданная экранная форма будет записана на диск.

Заметим, что если экранная форма ранее уже записывалась с помощью позиции **Save this form** меню **Layout**, то после выбора **Save changes and exit** меню **Exit** уже не будет высвечиваться окно с предложением ввести имя. При сохранении с помощью позиции меню **Layout** окно для ввода имени высвечивается всегда, что можно использовать для записи копии разработанной экранной формы под другим именем.

Запуск экранной формы из УЦ осуществляется нажатием **F2** после установки курсора на имя экранной формы или нажатием **Enter** и выбором позиции **Display data** в высветившемся горизонтальном меню. Причем если в УЦ открыт файл БД, не связанный с данной формой, то высвечивается горизонтальное меню с двумя позициями: **Current view** (текущая выборка) и имя файла БД или выборки, связанной с вызываемой формой.

В нашем учебном примере каждая экранная форма связана с определенным файлом БД, имя которого и высвечивается. Поэтому нужно перейти на это имя и нажать клавишу **Enter**.

Переход по полям в высветившейся экранной форме производится навигационными клавишами **Вверх**, **Вниз**, а переход на следующую и предыдущую запись клавишами **PgUp** и **PgDn**. В прямом направлении по полям можно также передвигаться клавишами **Tab** и **Enter**, а в обратном - клавишами **Shift-Tab**.

Заметим, что после перемещения курсора на **Memo** поле редактировать данные в нем еще нельзя. Режим редактирования **Memo** поля активизируется клавишами **Ctrl-Home**. Выход из режима редактирования осуществляется клавишами **Ctrl-End**. Для выхода из режима высвечивания экранной формы, нужно активизировать меню клавишей **F10**, в котором следует воспользоваться позицией **Exit**.

Для закрепления полученных навыков попробуйте самостоятельно создать экранные формы ввода и вывода данных из файлов **MBD2**, **MBD3**, **MBD4**.

В учебных примерах этим экранным формам присвоены имена **MF2, MF3** и **MF4**, а их вид показан на рис. 2.5-2.7.

```
Layout   Fields   Words   Go To   Exit                12:54:38 pm
Для выхода в меню нажмите клавишу F10

       Автомобиль: XXXXXXXXXX   номерной знак: XXXXXXXXX

       Техническое состояние:  XXXXXXXXXXXX

       Где находится в данное время: XXXXXXXXXXXX

Form  |C:\dbase\primer\MF2  |Row:0 Col:0  |File:MBD2 |      Ins
     Add field:F5  Select:F6  Move:F7  Copy:F8  Size:Shift-F7
```

Рис. 2.5. Экранная форма данных о состоянии автомобилей

```
Layout   Fields   Words   Go To   Exit                 2:55:16 pm
Для выхода в меню нажмите клавишу F10

   Наименование груза: XXXXXXXXXXXXXXXXXXXX    Вес - 99.9 т.

   Номерной знак автомобиля, перевозящего груз: XXXXXXXXX

   Исходный пункт: XXXXXXXXXXXXXXX     Дата  выезда: DD/MM/YY

   Конечный пункт: XXXXXXXXXXXXXXX   Дата прибытия: DD/MM/YY

   Расстояние: 9999 км      Время в пути (фактич.):  999 ч

   Расход горючего (фактич.):  9999.99

Form  |C:\dbase\primer\MF3 |Row:0 Col:0   |File:MBD3   |     Ins
    Add field:F5  Select:F6  Move:F7  Copy:F8  Size:Shift-F7
```

Рис. 2.6. Экранная форма данных о перевозках

```
Layout   Fields   Words   Go To   Exit                    12:55:48 pm
Для выхода в меню нажмите клавишу F10

              Технические характеристики

          Марка: XXXXXXXXXX

          Скорость: 999 км/ч

          Грузоподъемность: 99.9 т

          Расход топлива на 100 км : 999.99 л

Form  |C:\dbase\primer\MF4  |Row:0 Col:0   |File:MBD4  |     Ins
     Add field:F5  Select:F6  Move:F7  Copy:F8  Size:Shift-F7
```

Рис. 2.7. Экранная форма технических характеристик автомобилей

2.7. РАЗРАБОТКА ОТЧЕТОВ

Разработка отчетов наряду с экранными формами является завершающим этапом подготовки технологии обработки информации. При выдаче отчетов могут использоваться выборки данных, полученные в результате выполнения запросов, и исходные файлы БД. В итоге разработки отчета создается (генерируется) программный файл, который выполняется при выводе отчета. При выполнении программного файла отчета осуществляется извлечение данных, представление их в форме, требуемой для вывода на печать, и посылка символов и управляющих кодов на печатающее устройство.

Каждому отчету (программному файлу) присваивается имя, выбрав которое в окне отчетов на панели УЦ, можно распечатать отчет на принтере, вывести его на экран или в текстовый файл на диск. Программный файл отчета позволяет выполнять не только собственно операции по форматированию отчета, но и ряд предварительных операций по обработке данных. Например, можно задать условия для отбора требуемых записей файла БД в отчет и выполнить арифметические действия, как это делается с помощью запросов. Кроме этого, при подготовке отчетов можно выполнять более сложные операции, не предусмотренные в запросах: агрегирование, статистическую

информации и др. Таким образом, отчет является весьма мощным средством не только форматирования, но и многопланового обработки информации. В нем объединяются функции выполнения запросов на отбор информации, ее преобразования и представления в удобной форме для анализа.

Разработка отчета выполняется в окне **Reports** УЦ. Вход в экран разработки отчетов из УЦ осуществляется выбором позиции **<create>** в окне **Reports**. Предварительно необходимо активизировать файл БД или выборку, которые будут использоваться в отчете. Если требуется модифицировать уже созданный программный файл отчета, сначала надо выбрать его имя в окне **Reports** и затем позицию **Modify layout** в высветившемся меню. Короткий способ выхода на экран разработки отчета реализуется клавишами **Shift-F2** после предварительной установки курсора на имя отчета.

Форма отчета создается на экране. В нее можно включать следующие элементы: поля из текущего файла БД или выборки, вычисляемые поля, которые создаются специально для отчета, текстовый материал, рамки и линии. Текст длинных символьных полей можно размещать и разворачивать в колонках заданной ширины.

Экран разработки отчета состоит из горизонтальных полос. Эти полосы разделяют отчет на составные части. В число стандартных частей входят: **заголовок страницы, введение (заголовок) отчета, тело отчета, итоги (заключение) отчета** и **подножие страницы**. Тело отчета формируется в одной полосе. Остальные части отчета представляются в экранной форме парами: заголовок страницы - подножие страницы, введение отчета - итоги отчета.

Кроме указанных элементов можно добавлять полосы, объединяющие записи в группы по номеру записи, значению поля или по выражению, включающему поля файла БД, вычисляемые поля и специальные поля, функции и операторы. Например, в файле адресов места жительства можно объединять каждые **N** записей, создавать группы записей лиц с одинаковым местом жительства или группы с одинаковыми первыми тремя цифрами почтового индекса. Каждая группа включает пару полос - введение (заголовок) группы и итоги (заключение) группы.

Группирование записей обычно применяется для отсортированных или индексированных файлов БД. Группирование часто используется для получения промежуточных итогов, вычисления статистических функций: усреднения, подсчета количества записей, определения максимального и минимального значений, подсчета сумм, расчета среднеквадратического отклонения и дисперсии.

При разработке отчета используются два режима. Режим развертки текста и режим размещения элементов отчета, которые включают поля, рамки, линии и другие оформительские

элементы (для краткости этот режим будем называть режимом размещения полей).

Первый режим похож на обычный текстовый редактор и позволяет вводить текст непрерывной строкой, который разворачивается по вертикали в заданных границах. При правках текста в этом режиме на место удаленного слова или предложения будет подтягиваться текст, расположенный справа, а вставленное слово или предложение будет раздвигать текст вправо. Этот режим может использоваться для создания введения или заключения к отчету и ввода других текстовых материалов.

Режим размещения полей позволяет расположить в нужные места на экране разработки отчета обычные и вычисляемые поля файла БД. В этом режиме также можно вводить текст, но при удалении или вставке слова или предложения текст справа не передвигается. Признаком режима размещения полей на экране служит мелкая сетка. Если экран чистый, то это признак режима развертки текста. Мелкая сетка на экране в режиме размещения элементов является не только признаком режима, но и несет, как и при разработке экранных форм, служебную функцию - обозначает, что экран пустой (на нем нет ни символов, ни пробелов). Ввод пробелов приводит к очистке от сетки области экрана. Как и при выводе полей в экранной форме, вывод на печать полей, находящихся в очищенной области, отличается от вывода на печать, когда они разделены мелкой сеткой. В первом случае, при выводе на печать значения полей прижимаются друг к другу (между ними оставляется только один пробел), а во втором они остаются в фиксированных положениях в соответствии с длиной полей, определенных в файле БД. Для перехода в режим развертки текста нужно выбрать значение **Yes** в позиции **Word wrap band** меню **Bands**, а для выхода из него - значение **No**.

В верхней части экрана разработки отчетов имеется семь меню: **Layout**, **Fields**, **Bands**, **Words**, **Goto**, **Print** и **Exit**. Меню **Layout** предназначено для быстрого размещения на экране разработки отчета всех полей файла БД в стандартных формах, рисования рамок и линий, ввода словесного описания отчета, присоединения к отчету файла БД, отличного от активного в текущий момент, и сохранения программного файла отчета на диске.

Для ввода в форму отчета только требуемых полей используется меню **Fields**. Меню **Bands** позволяет создавать и удалять полосы, предназначенные для вывода групп данных, а также назначать каждой полосе свои опции печати.

С помощью меню **Words** и **Goto** можно позиционировать текст, добавлять и удалять строки, вводить текст из внешних текстовых файлов, передвигаться по разрабатываемой форме отчета, замещать слова и т.д.

Меню **Print** предназначено для установки опций печати выходного документа в целом, указания вывода отчета на принтер, экран, внешний текстовый файл, установки посылки на принтер управляющих кодов и т.д., а меню **Exit** для выхода из окна разработки отчета.

Если в форму отчета должны входить группы, то целесообразно начать разработку с определения состава групп, для чего предназначено меню **Bands**. Для ввода на экране пары полос группы выбирается позиция **Add group band** этого меню. Предварительно курсор должен быть установлен выше тела отчета на полосе заголовка страницы (**Intro Page Band**), заголовка отчета (**Intro Report Band**) или на полосе уже существующей группы (**Group N Intro Band**, где **N** - номер группы). После этого нужно выбрать тип группы из высветившегося меню: по значению поля - **Field value**, по выражению - **Expression value** и по количеству записей - **Record count**.

При выборе позиции **Field value** высвечивается список полей из файла БД, а также вычисляемые поля, ранее определенные для данного отчета. Выбранное поле используется в качестве группирующего признака.

При выборе типа группы **Expression value** требуется ввести выражение для определения группы. Можно использовать клавиши **Shift-F1** для выбора имен полей, операторов или функций из высвечиваемого списка, а также вводить выражение непосредственно с клавиатуры.

При выборе типа группы **Record count** требуется ввести число записей, определяющих группу.

В отчет можно ввести несколько групп, которые будут вложенными. Созданные полосы групп нумеруются. Нумерация групп соответствует уровню вложенности (не зависит от порядка создания группы, а только от ее положения по отношению к другим группам). При печати отчета с группами сначала выводится внешняя группа данных (с номером 1). Внутри этой группы выводятся группы следующего уровня вложения и т.д. Тип группы высвечивается внизу экрана при установке курсора на соответствующую полосу.

Для удаления группы используется позиция **Remove group**. При этом удаляется текущая группа, на которую установлен курсор, включая обе полосы, входящие в состав группы - введение группы и ее итоги. Группы, вложенные внутри текущей группы, не удаляются. Для удаления можно также использовать клавишу **Del**.

Если требуется изменить тип группы, то используется позиция **Modify group**. При этом высвечивается меню типов группы, что позволяет изменять тип группы или значения (выражения), использованные для формирования групп.

При формировании отчетов с большим числом групп для удобства работы может потребоваться скрыть содержимое всех

групп, за исключением текущей разрабатываемой группы. Содержимое группы скрывается клавишей **Enter** при установке курсора на верхнюю полосу группы. Для того чтобы быстро открыть содержимое всех групп, используется позиция **Open all bands**.

Остальные позиции меню **Bands** позволяют установить опции печати специально для текущей группы. Выбор позиции **Group intro on each page** дает возможность выводить заголовок группы на каждой странице, пока продолжается данная группа. Эта позиция используется, например, для вывода заголовка длинных таблиц, печатаемых на нескольких страницах.

Для печати новой группы с начала страницы применяется позиция **Begin band on new page**. Позиция **Text pitch for band** позволяет выбрать тип шрифта: **PICA** (обычный - 10 символов на дюйм), **ELITE** (уменьшенный - 12 символов на дюйм), **CONDENSED** (сжатый), **DEFAULT** (установленный в меню **Print**). Качество печати устанавливается позицией **Quality print for band**: качественная - **YES**, обычная - **NO** или установленная в меню **Print** - **DEFAULT**. Межстрочный интервал выбирается позицией **Spacing of lines for band**: **SINGLE** (через одну строку), **DOUBLE** (через две строки), **TRIPLE** (через три строки), **DEFAULT** (установленный в меню **Print**).

Для того чтобы не печатать заголовок и подножие страницы, во введении и итогах отчета выбирается значение **No** в позиции **Page headings in report intro**.

После формирования групп в отчете можно вводить поля. Для ввода полей можно использовать позицию **Quick layout**, позволяющую ввести все поля файла БД в виде трех стандартных типов отчетов:

типа экранной формы **Browse** (**Column layout**) - все поля БД располагаются горизонтально до 255 позиций в ширину. Если все поля не умещаются, то высвечивается сообщение об ошибке и оставшиеся поля отбрасываются;

типа экранной формы **Edit** (**Form layout**) - поля располагаются по вертикали. Полоса введения отчета закрывается. Номер страницы и дата располагаются в полосе заголовка страницы;

типа письма (**Mailmerge layout**) - все полосы за исключением тела отчета закрываются. Устанавливается режим развертки текста и печати полосы на новой странице, что позволяет вводить текстовый материал и печатать каждую запись файла БД на новой странице.

Поля можно перемещать и копировать на экране с помощью клавиш **F7** и **F8**. Предварительно следует установить курсор в поле и высветить его клавишей **F6**. После этого нужно переместить курсор на новое место и нажать клавишу **F7** (**F8**) для перемещения (копирования) поля и затем зафиксировать перемещение клавишей **Enter**. Размер поля можно изменить клавишами **Shift-F7** также после высвечивания поля. После

изменения размера навигационными клавишами **Вправо** и **Влево** следует также зафиксировать новый размер клавишей **Enter**.

Основным способом ввода в форму отчета всех типов полей служит позиция **Add field** меню **Fields** или для более короткого ввода - клавиша **F5**. После этого высвечивается таблица, состоящая из четырех колонок. В первой колонке содержатся имена полей текущего файла БД (выборки). В случае использования выборки в число полей включаются и вычисляемые поля, созданные в этой выборке. Вторая колонка содержит имена новых вычисляемых полей, которые включаются в отчет. Третья колонка содержит имена системных полей: **Date** (текущая дата), **Time** (текущее время), **Recno** (номер записи), и **Pageno** (страница), которые могут использоваться в отчете. Четвертая колонка содержит имена операторов агрегирования и статистических функций: **Average** (среднее значение), **Count** (количество записей), **Max** (максимальное значение), **Min** (минимальное значение), **Sum** (сумма), **Std** (среднеквадратическое отклонение) и **Var** (дисперсия). Эти операторы используются для создания полей в полосах итогов (подытогов) отчета, в которые помещаются результаты агрегирования.

После выбора имени поля файла БД в таблице высвечивается меню, в котором описано это поле и содержатся три позиции: **Template** - для ввода знаков шаблона поля, **Picture function** - для установки функций поля, **Supress repeated values YES (NO)** - для подавления (вывода) повторяющихся значений поля. Знаки шаблона для различных типов полей, применяемые при разработке отчетов, ничем не отличаются от используемых для экранных форм и описаны в разделе разработки экранных форм. Функции поля, предназначенные для печати, несколько отличаются от функций, применяемых для экранных форм. Для числовых полей используются следующие функции:

C - аббревиатура CR (кредит) после положительного числа;
X - аббревиатура DB (дебет) после отрицательного числа;
(- отрицательное число в круглых скобках;
L - печать ведущих нулей числа;
Z - пробелы вместо ведущих нулей числа;
$ - знак валюты впереди числа;
^ - формат числа с порядком (научная запись числа);
T - удаление ведущих и конечных пробелов;
B - выравнивание влево;
I - центрирование;
V - вертикальная развертка по шаблону поля.

Для символьных полей применяются функции:

! - преобразование строчных букв в прописные;
R - печать заданных символов;
T - удаление ведущих и конечных пробелов;
J - выравнивание вправо;
I - центрирование;

V - вертикальная развертка по шаблону поля;

; - вертикальная развертка по признаку.

Как и при разработке экранной формы, функции поля устанавливаются выбором значения **Yes** для соответствующей функции. После установки требуемых знаков шаблонов и функций поля их необходимо сохранить клавишами **Ctrl-End**.

Для создания нового вычисляемого поля требуется выбрать позицию **<create>** в верхней части таблицы. После этого высвечивается меню, в позиции которого следует ввести с клавиатуры: имя вычисляемого поля (**Name**), словесное описание вычисляемого поля (**Description**), выражение, определяющее вычисляемое поле (**Expression**). При записи выражения вычисляемого поля можно выбирать исходные поля файла БД, ранее созданные вычисляемые поля, операторы и функции из списка, высвечиваемого клавишами **Shift-F1**, или же вводить выражение непосредственно с клавиатуры. В выражениях нельзя использовать системные поля типа даты (**Date**) и др. После установки значений всех позиций требуется их сохранить клавишами **Ctrl-End**.

Вычисляемое поле можно скрыть и не печатать (например, при использовании его только в составе выражения или в других целях) выбором значения **Yes** в позиции **Hidden**.

Для ввода системного поля, типа текущей даты и времени его имя выбирается из таблицы. При этом высвечивается меню, в позициях которого можно задать установку (**Yes**) на подавление повторяющихся значений (**Suppres repeated values**) или изменить клавишей **Enter** имя системной переменной (**Name**).

При создании в отчете поля, в которое заносятся результаты агрегирования, оператор агрегирования выбирается из таблицы, затем в высветившееся меню после выбора позиции **Name** вводится имя этого поля. Можно также ввести словесное описание выполняемой операции (позиция **Description**), поменять оператор агрегирования (если это требуется) посредством выбора позиции **Operation** и затем выбрать с помощью позиции **Field to summarize on** требуемые поля файла БД из высвечиваемого списка, с которыми выполняется операция агрегирования (кроме подсчета числа записей), а также с помощью позиции **Reset every** установить сброс в ноль значения текущего поля агрегированной величины после каждой страницы (опция **Page**) или в конце отчета (опция **Report**). Знаки шаблона и функции полей агрегированных величин полностью аналогичны числовым и вычисляемым полям. Операторы агрегирования следует помещать в тело отчета или в итоговые полосы групп.

Если в процессе формирования отчета требуется удалить какое-либо поле из формы, это можно сделать выбором позиции **Remove field** или клавишей **Del** после установки курсора в удаляемое поле.

Модификация введенного в отчет поля выполняется выбором позиции **Modify field** (на экране появляется таблица полей, как и при их вводе в отчет) или при установленном на поле курсоре клавишами **Ctrl-Home** или **F5**.

Для того чтобы восстановить скрытое вычисляемое поле или поле агрегированной величины (или скрыть его), это поле нужно выбрать из списка, высвечиваемого выбором позиции **Change hidden field**. При восстановлении поле размещается в текущей позиции курсора.

После того как введены все требуемые поля, можно вводить в форму отчета заголовки и подножия страниц, введение и заключение и т.д., которые содержат текстовой материал и другие оформительские элементы.

Для рисования рамок и линий используются позиции **Box** и **Line** меню **Layout**. Их применение описано в разделе разработки экранных форм.

При оформлении полос отчета можно использовать меню **Words** и **Goto**. Эти меню общие как для разработки отчетов, так и для экранных форм. Меню **Goto** полностью описано в разделе 2.6, а меню **Words** - за исключением позиций, которые там не используются. Эти позиции относятся только к разработке форм отчетов. Ниже будут описаны эти позиции, а также некоторые из используемых и в экранной форме, но имеющие специфику применения в двух режимах разработки формы отчета. Следует отметить, что для разработки форм отчета, в свою очередь, не используется позиция **Display**, которая служит для раскраски областей экрана.

К специфическим позициям для разработки форм отчетов относится позиция **Style**, служащая для формирования стиля печати. После выбора этой позиции высвечивается меню с типами шрифтов, которые можно установить для выделенных областей текста: **Normal** - нормальный, **Bold** - жирный, **Underline** - подчеркнутый, **Italic** - наклонный, **Raised** - приподнятый, **Lowered** - опущенный.

Позиция **Enable automatic indent ON (OFF)** включает (выключает) автоматический абзац в режиме развертки текста. При выборе **ON** действие клавиш **Tab** и **Shift-Tab** в том случае, если курсор находится в начале параграфа, приводит к переустановке левой границы. Если установлено **OFF**, то нажатие клавиш **Shift-Tab** передвигает курсор к следующей позиции табуляции. Действие клавиши **Tab** зависит от установки на клавиатуре режима "вставки символов" клавишей **Ins**. Если режим "вставки символов" выключен, то клавишей **Tab** курсор передвигается к следующей позиции табуляции. Если режим "вставки" включен, то клавиша **Tab** вводит в разметку текстового редактора символ табуляции, который передвигает курсор к следующей позиции табуляции. В режиме развертки текста при передвижении левой границы абзац всегда остается в том же относительном положе-

нии. Следует отметить, что позиция этого меню **Position**, служащая для позиционирования текста, действует только в режиме размещения полей.

Позиция **Insert page break** устанавливает прерывание печати страницы на текущей линии в режиме развертки текста, в режиме размещения полей она не действует.

Позиция **Modify ruler** используется в режиме разработки как экранных форм, так и отчетов и служит для изменения разметки экрана. Эта разметка высвечивается в режиме размещения полей и в режиме развертки текста. При разработке экранных форм применяется только режим размещения полей. Разметка высвечивается в верхней части экрана и включает символы, обозначающие левую и правую границы ([и]), абзац (#), табуляцию (треугольник) и номер колонки.

Движение по разметке осуществляется курсором или клавишей **Пробел**. Движение вправо и влево по восемь позиций можно выполнить клавишами **Ctrl-вправо** и **Ctrl-влево**, а по символам табуляции - клавишами **Tab** и **Shift-Tab**. Клавиши **End** и **Home** передвигают курсор в левую и правую видимую часть экрана. При повторном нажатии этих клавиш происходит передвижение еще на одну секцию разметки. Установка символов разметки выполняется соответствующими клавишами, кроме знака табуляции, который вводится клавишей !. Удаление символов разметки выполняется как в обычном режиме редактирования текста: клавишами **Del** и **Backspace**. Если в одной позиции размещается несколько символов, то высвечивается старший по приоритету в соответствии со следующей последовательностью (по убыванию приоритета): граница, абзац, табуляция и номер колонки. Сохранение установок выполняется клавишами **Ctrl-End**. Устанавливаемая позицией **Modify ruler** разметка не сохраняется при вызове редактора для редактирования **Memo** полей, программ или текстовых файлов.

Рассмотрим более подробно действие позиции **Add line**, служащей для добавления новой строки после текущей. В режиме развертки текста сначала закрывается текущий параграф и курсор передвигается в начало новой строки.

В режиме размещения полей добавляется новая строка в экранную форму и курсор устанавливается в начало этой строки. Эта позиция не используется при разработке экранной формы марки, так как марка имеет фиксированную высоту и, следовательно, в нее нельзя добавлять новых строк (см. раздел 2.8).

Позиция **Remove line**, служащая для удаления текущей строки, в режиме развертки текста заменяет текст текущей строки пробелами, но не удаляет строку, а в режиме размещения полей вместе с текстом из формы удаляется и строка.

Если к разрабатываемой форме отчета требуется присоединить другой файл БД или выборку, то для этого следует выбрать позицию **Use different database file or view** меню **Layout**.

Словесное описание отчета, которое высвечивается при его выборе на панели УЦ, можно ввести с помощью позиции **Edit description of report** этого же меню. Для сохранения созданного отчета (программного файла) на диске с присвоением ему имени используются такие же позиции, как и для экранных форм: **Save this report** - сохранение без выхода из экрана разработки, **Exit** - выход с сохранением или без.

Меню **Print** позволяет выполнить ряд стандартных действий по управлению принтером во время печати документа, включая запуск отчета на печать или запись отчета в текстовой файл **DOS**.

После того как отчет сформирован, необходимо установить формат печатаемой страницы. Для этого служит позиция **Page dimensions**. После ее выбора появляется меню с опциями: установка размера страницы (**Length of page**), установка сдвига печати слева (**Offset from left**), установка интервала между строками (**Spacing of lines**) со значениями **SINGLE** - одна строка, **DOUBLE** - две строки, **TRIPLE** - три строки).

Нередко требуется напечатать не весь документ, а только его часть (или повторить печать нескольких страниц). Для этого используется позиция **Output options**. В меню этой позиции имеются опции: начать печать со страницы с **N** (**N** - порядковый номер страницы) (**Begin on page**), окончить печать на странице с номером **N** (**End after page**), установить начальный номер на первой печатаемой странице (**First page number**). При этом номер страницы будет печататься только в том случае, если задана функция печати номера страницы (**Pageno**).

Для установки этой функции нужно выбрать в меню **Fields** основного (горизонтального) меню разработки отчета позицию **Add fields**, а затем в высветившейся таблице установить курсор на позицию **Pageno** колонки **PREDEFINED** и нажать клавишу **Enter**. После этого высветится окно, в котором показан шаблон поля номера страницы, заданный по умолчанию (**Template**), позиция для задания функций поля (**Picture functions**) и опция для подавления повторяющихся значений полей (**Suppres repeated values YES/NO**). В документах номер страницы обычно окружается знаками "тире". Чтобы эти знаки печатались, нужно поставить их в шаблоне, который будет выглядеть, например, так: {- 99 -}. Для фиксации изменения шаблона нужно нажать **Enter**, а затем сохранить все значения клавишами **Ctrl-End**.

Возвращаясь к меню **Print**, укажем, что для распечатки нескольких копий документа служит опция **Number of copies**. Отметим, что в качестве начального и конечного номеров печатаемых страниц при перепечатке надо указывать не физические номера страниц, а их порядковые номера, соответствующие принятой нумерации, начиная с заданного номера на первой физической странице.

Позиция **Control of printer** позволяет направить на принтер команды (управляющие коды) для управления его действиями.

Если подача бумаги на принтере не автоматическая, то обычно требуется установить режим приостановки принтера после печати страницы для заправки новой страницы. Для этого используется опция **Wait between pages**. На принтерах обычно имеются переключатели шага печати - **PICA, ELITE, CONDENSED**.

Опция **Text pitch** позволяет установить один из них дистанционно, что удобно, если принтер стоит далеко от ПК. В этой опции имеется также значение **DEFAULT**, служащее для использования шага печати, установленного на принтере. Можно также дистанционно установить качественную печать (опция **Quality print** - значения **YES, NO** и **DEFAULT**), направить на принтер стартовые и финишные управляющие коды - опции **Starting control codes** и **Ending control codes**, установить режим подачи бумаги - опция **Advance page using** со значениями: **LINE FEED** (построчно), **FORM FEED** (постранично) и установить режим перехода к новой странице - опция **New page** со значениями: **BEFORE** (переход к новой странице до печати отчета), **BOTH** (переход к новой странице до и после печати отчета), **NONE** (без перехода к новой странице), **AFTER** (переход к новой странице после печати отчета).

Позиция **Destination** позволяет вывести отчет на принтер или в текстовой файл **DOS** в зависимости от установки опций **Write to printer/DOS File** (переключение выполняется клавишей **Enter**). Если отчет нужно направить в файл **DOS**, тогда требуется присвоить ему имя в опции **Name of DOS File**. Опция **Printer model** служит для установки типа принтера: **GENERIC** или **ASCII TEXT**. Эта опция используется и в том случае, если отчет направляется в файл **DOS**, так как в этот файл включаются управляющие коды того принтера, на котором впоследствии предполагается печатать файл **DOS**. Значение **GENERIC** используется при создании файла, включающего наиболее общие управляющие коды принтера, а **ASCII TEXT** - для создания текстовых файлов. Опция **Echo to screen** позволяет высвечивать печатаемый отчет на экране.

Иногда требуется перевести страницу на принтере не непосредственно перед печатью отчета, а заранее. При выборе позиции **Eject page now** страница переводится немедленно.

Все сделанные установки для принтера целесообразно записать в специальный файл установок печати выбором позиции **Save settings to print form**. После этого высвечивается окно **Enter name of print file** с текущим маршрутом по директориям и именем отчета, если до этого отчет уже был сохранен на диске с помощью позиции **Save this report** меню **Layout**. Это имя можно принять для файла установок или дать свое имя. Сохраненный файл установок печати отчета будет автоматически вызываться при печати отчета. Кроме печати данного отчета, этот файл можно использовать и для печати других отчетов. Для

этого его нужно выбрать из списка файлов этого типа, высвечиваемых после выбора позиции **Use print form**.

Заметим, что для печати кириллицей перед записью параметров печати нужно назначить тот тип принтера, который был задан при установке пакета dBASE IV. Обычно к этому принтеру подключается драйвер, обеспечивающий печать кириллицы. Это назначение выполняется клавишами **Enter** или **Пробел** после выбора позиции **Printer model** подменю (позиции) **Destination** меню **Print**. Перед печатью отчет можно сначала вывести на экран с целью контроля правильности его формирования. Для этого служит позиция **View report on screen**. Собственно запуск отчета на печать выполняется выбором позиции **Begin printing**.

Покажем теперь, как создать программный файл отчета с именем **MR1**, генерирующий ведомость расхода горючего водителями за октябрь 1989 года, представленную ниже.

Ведомость содержит планируемый и фактический расход горючего каждым водителем за октябрь, перерасход или экономию горючего водителем, а также общий перерасход или экономию горючего на автопредприятии за этот месяц. В конце страницы печатается номер ведомости.

Чтобы получить такую ведомость, нужно отобрать данные по выполненным водителями в октябре перевозкам. Для этого предварительно нужно составить запрос на связку файлов **MBD1, MBD2, MBD3, MBD4** по общим полям (**MBD1-MBD2-MBD3** по полю **N_Z**, **MBD2-MBD4** по полю **MAR**) и выполнение отбора данных по перевозкам, отмеченных октябрем.

Р А С Х О Д
горючего за октябрь 1989 г.

Пункты перевозки груза		Расстояние км	Расход горючего, л	
Откуда	Куда		планируемый	фактич.
Водитель: Ермишкин Ю.И.	Автомобиль: ЗИЛ-131	78-99 ММС		
Москва Болшево	Ярославль Тула	300 200	51.00 34.00	50.00 33.00
Перерасход (-), экономия (+):		2.00	85.00	83.00
Водитель: Садчиков А.В.	Автомобиль: ЗИЛ-131	99-99 ММД		
Москва Москва	Горький Подольск	400 [illegible]	68.00 [illegible]	69.00 [illegible]
Перерасход (-), экономия (+):		-2.20	74.80	77.00
Итого расход топлива: 160.00 Перерасход (-), экономия (+): -0.20				

Ведомость N 35

Этот запрос формируется в окне **Queries** УЦ подобно запросу на связку файлов **MQ15**, описанному в разделе создания запросов, а в качестве критерия поиска в файле **MBD3** в поле **D_V** (дата выезда) нужно ввести условие

> = {01.10.89}, < = {31.10.89}

согласно которому будут отбираться записи по перевозкам, выполненным в октябре и дадим этому запросу имя **MQ16**. Напомним, что в формируемый шаблон вывода для связки файлов надо включать все поля, которые будут использоваться в отчете.

Как уже упоминалось в разделе формирования запросов, связку файлов можно выполнить с помощью меню или вручную с клавиатуры. В качестве упражнения создайте вручную с клавиатуры запрос аналогичный **MQ16**, помещая в графы общих полей какие-либо имена, например, для поля **N_Z** имя **L1**, а для поля **MAR** - имя **L2**, и запишите его под именем **MQ17**. Для вывода на печать данных о всех перевозках нужно в запросе **MQ17** (или **MQ16**) удалить критерий отбора. Запишите этот запрос под именем **MQ18**.

Перед созданием отчета следует активизировать один из запросов **MQ16, MQ17** или **MQ18** - выбрать его в окне **Queries** панели УЦ. Создание отчета начинается выбором позиции **<create>** в окне **Reports** УЦ. После этого высвечиваются стандартные полосы, из которых составляется отчет. Сверху помещается полоса заголовка страницы (**Page Header Band**), ниже полоса введения в отчет (**Report Intro Band**). Это означает, что заголовок страницы будет печататься и во введении в отчет. Чтобы избежать этого, выберите значение **NO** позиции **Page heading in report intro** меню **Bands**. После этого полоса **Report Intro Band** переместится вверх.

Поскольку в ведомости надо вывести группы данных по выполненным каждым водителем перевозкам, то установите группирование записей по полю **FAM** (фамилия). В этом случае все отобранные по каждому водителю записи из связанных файлов БД будут подобраны в отдельные группы, которые и составят содержание ведомости.

Для группирования записей по полю **FAM** выберите позицию **Add a group band**, а затем позицию **Field value** меню **Bands** и в высветившемся окне из списка полей - поле **FAM**. После этого на экране появятся еще две полосы **Group 1 Intro Band** и **Group 1 Summary Band**.

Разработку частей отчета целесообразно начать с ввода текста введения в отчет (полоса **Report Intro Band**). В нашем случае введением в отчет будет заголовок ведомости. Для этого сначала введите две пустые строки в эту полосу с помощью позиции **Add line** меню **Words** или нажав два раза клавишу **Enter**, а затем текст заголовка.

```
— Report Intro Band ————————————————————————————————————————————
                          Р А С Х О Д
                   горючего за октябрь 1989 г.
— Page Header Band —————————————————————————————————————————————
```

В полосу заголовка страницы (**Page Header Band**) введите заголовок страницы, который будет печататься вверху каждой страницы отчета. Линии вычерчиваются с помощью позиции **Line** меню **Layout** подобно тому, как это описано в разд. 2.6.

Следует отметить, что для соединения при выводе ведомости вертикальных линий заголовка с вертикальной разграфкой тела отчета, нужно в заголовке для этих вертикальных линий сделать отводы вниз, как показано ниже. Это же относится к соединению вертикальных линий разграфки тела отчета с подножием страницы и другими частями отчета, где такие отводы надо делать вверх или вниз.

```
— Page Header Band —————————————————————————————————————————————
┌──────────────────────────────────┬───────────┬────────────────────────┐
│    Пункты перевозки груза        │Расстояние │  Расход горючего, л    │
├───────────────┬──────────────────┤    км     ├────────────┬───────────┤
│    Откуда     │       Куда       │           │планируемый │ фактич.   │
╞═══════════════╧══════════════════╧═══════════╧════════════╧═══════════╡
— Group 1 Intro Band ———————————————————————————————————————————
```

Теперь приступим к формированию полосы заголовка группы **Group 1 Intro Band**. Введите в нее поля **FAM**, **NAM**, **OTC**, **MAR**, **N _ Z**, расположив их следующим образом:

```
— Group 1 Intro Band ———————————————————————————————————————————
│ Водитель: XXXXXXXXXXXXXX X.X. Автомобиль: XXXXXXXXX XXXXXXXX   │
├─────────────────┬─────────────────┬───────────┬───────────┬────┤
— Detail          Band —————————————————————————————————————————
                  FAM      NAM OTC                MAR        N_Z
```

Ввод полей осуществляется выбором позиции **Add field** меню **Fields** или более коротким способом - клавишей **F5**. Предварительно нужно установить курсор на ту позицию в полосе, где должно размещаться поле.

Поясним более подробно процесс ввода поля **FAM**. Сначала целесообразно ввести текстовую часть перед этим полем - "Водитель:". Затем установите курсор после этого слова и нажмите клавишу **F5**. Высветится таблица с четырьмя колонками **MQ16** (перечень имен полей активной выборки, в данном случае с именем **MQ16**), **CALCULATED** (перечень имен вычисляемых полей), **PREDEFINED** (системные поля), **SUMMARY** (суммируемые поля). Из списка полей в колонке **MQ16** выберите поле **FAM**. После этого на экране появится окно для установки параметров поля с позициями **Template** (шаблон поля), **Picture functions** (функции поля) и **Suppress repeated values** (подавить поворяющиеся значения поля). Примем установленные

по умолчанию значения этих позиций для поля **FAM**, для чего надо нажать клавиши **Ctrl-End**.

После поля **FAM** введите последовательно поля **NAM** (имя), **OTC** (отчество), **MAR** (марка) и **N_Z** (номерной знак). Заметим, что при вводе полей **NAM** и **OTC** их шаблоны нужно представить в виде **"X."**, чтобы выводить только инициалы водителя с точками. Изменение шаблона поля выполняется в следующей последовательности. Сначала установите курсор на поле, затем нажмите клавишу **F5** для вызова окна параметров этого поля, затем выберите в этом окне позицию **Template** и нажмите клавишу **Enter**, после чего курсор установится в шаблоне и его можно изменять (редактировать) как обычное поле посредством ввода или удаления требуемых знаков с клавиатуры.

В тело отчета (полоса **Detail Band**) нужно ввести поля, включающие данные о перевозках, выполненных водителем. Это поля **O_P** (исходный пункт), **K_D** (конечный пункт), **RAS** (расстояние), **RGR** (расход горючего фактический) и вычисляемое поле **RGP**, определяющее планируемый расход горючего. Ввод всех полей кроме RGP осуществляется аналогично полям в полосе **Group 1 Intro Band**. Поясним более подробно процесс создания вычисляемого поля. Значение поля **RGP** вычисляется по формуле

RGP = RSX*RAS/100

где **RSX** - расход горючего в литрах на 100 км, определенный в технических характеристиках автомобиля, а **RAS** - пройденное автомобилем расстояние.

Для создания вычисляемого поля **RGP** сначала переместите курсор в позицию, где нужно разместить это поле и нажмите клавишу **F5** (напомним, что действие этой клавиши эквивалентно выбору позиции **Add field** меню **Fields**), после чего высветится уже известная таблица с четырьмя колонками: **MQ16** (перечень имен полей активной выборки), **CALCULATED** (перечень имен вычисляемых полей), **PREDEFINED** (системные поля), **SUMMARY** (суммируемые поля). Для создания вычисляемого поля установите курсор на позицию **<create>** колонки **CALCULATED** и нажмите клавишу **Enter**, после чего высветится окно для установки параметров вычисляемого поля. Введите имя вычисляемого поля **RGP** и его описание, например, "Планируемый расход горючего", выбрав позиции **Name** и **Description**. В позицию **Expression** введите арифметическое выражение **RAS*RSX/100**, по которому будет вычисляться значение поля **RGP**. По умолчанию в позиции **Template** высвечивается шаблон поля, равный длине поля, определенной при создании структуры файла БД. Чтобы не выводить лишние позиции, сократите этот шаблон (клавишами **Del** или **Backspace**), оставив четыре знака до точки и два после (**9999.99**). После установки всех параметров вычисляемого поля

нажмите клавиши **Ctrl-End**. Вычисляемое поле **RGP** создано и введено в тело отчета. После этого тело отчета выглядит так:

```
-- Detail  Band ------------------------------------------------
| XXXXXXXXXXXX  | XXXXXXXXXXXXXX |   9999   | 9999.99   |  9999.99 |
-- Group 1 Summary Band-----------------------------------------
      O_P              K_D          RAS       RGP          RGR
```

Перейдем теперь к полосе подножия группы (**Group 1 Summary Band**), в которую нужно ввести три вычисляемых поля для расчета общего планируемого (**SRP**) и фактического (**SRR**) расхода горючего каждым водителем за октябрь, а также поле для расчета перерасхода или экономии горючего (**SRP-SRR**) этим водителем.

Сначала создадим поле **SRR**. Для этого установите курсор на нужную позицию в полосу **Group 1 Summary Band**, нажмите клавишу **F5**, после чего высветится уже известная таблица и теперь для создания суммируемого поля в колонке **SUMMARY** выберите позицию **Sum**. После этого высветится окно для установки параметров суммируемого поля. Введите имя вычисляемого поля - **SRR** и его описание - "Общий фактический расход", выбрав соответственно позиции **NAME** и **Description**. Далее надо указать поле, по которому будет выполняться суммирование и определяться общий фактический расход горючего. Это поле **RGR** выборки **MQ16**. Для этого установите курсор на позицию **Field to summarize on**, затем из высветившегося окна с именами полей активной выборки **MQ16** выберите поле **RGR**. Затем выберите также позицию **Reset every**, после чего высветится окно с позициями **<REPORT>**, **<PAGE>**, **FAM**. Первые две позиции позволяют выводить в поле **SRR** сумму значений поля **RGR** соответственно в конце отчета или страницы, а последняя позиция, в которой стоит поле **FAM**, после изменения фамилии. Установите курсор на позицию **FAM** и нажмите клавишу **Enter**. Это значит, что суммирование значений поля **RGR** будет выполняться только внутри группы 1, т.е. будет суммироваться горючее, израсходованное каждым водителем на перевозках грузов за октябрь, и выдаваться в подножии группы 1. Чтобы не выводить лишние позиции измените, как и при создании поля **RGP** шаблон поля **SRR** в позиции **Template** на **"9999.99"**. Для сохранения выполненных установок по определению поля нажмите клавиши **Ctrl-End**. Создание суммируемого поля **SRP** выполняется аналогично суммируемому полю **SRR** с той лишь разницей, что суммирование будет выполняться по вычисляемому полю **RGP**, имя которого находится в колонке **CALCULATED** таблицы, высвечиваемой после нажатия клавиши **F5**. Предварительно не забудьте переместить курсор на требуемую позицию в полосе **Group 1 Summary Band**.

Вычисляемое поле разности **SRP-SRR** в принципе создается подобно другим вычисляемым полям, например, полю **RGP**.

Различие состоит в назначении выражения вычисляемого поля при выборе позиции **Expression**. В данном случае это будет выражение **SRP-SRR**. Отметим, что до сих пор вычисляемым полям присваивалось уникальное имя посредством его ввода с клавиатуры при выборе позиции **Name** окна установки параметров. В принципе это можно не делать. Тогда само выражение будет служить его именем. Это можно продемонстрировать на примере поля разности **SRP-SRR**. При заполнении окна установок этого поля введите в позицию **Expression** выражение **SRP-SRR** и пропустите позицию **Name** (в позиции **Description** можно ввести описание этого выражения), сохранив заданные установки клавишами **Ctrl-End**. Предварительно не забудьте установить курсор на позицию, где будет размещаться это поле. После окончания формирования полей группы 1 можно обвести их рамкой, как показано ниже.

```
— Group 1 Summary Band ——————————————————————————————————————————
|                                                               |
| Перерасход (-), экономия (+):  999.99   |  999.99  |  999.99  |
|===============================================================|
— Page  Footer  Band ————————————————————————————————————————————
                                SPR-SRR       SPR       SRR
```

В подножии страницы (полоса **Page Footer Band**) можно поместить номер печатаемой ведомости, как показано ниже.

```
— Page  Footer  Band ————————————————————————————————————————————
                                          Ведомость   N 35
— Report Summary  Band———————————————————————————————————————————
```

Перейдем к полосе итогов отчета (**Report Summary Band**). В конце отчета нужно подсчитать, сколько всего было истрачено горючего на перевозках грузов автопредприятием за октябрь 1989 г., а также перерасход (экономию) горючего. Для вычисления этих показателей создайте поля **SAP** (общий планируемый расход горючего), **SAR** (общий фактический расход горючего), а также поле разности полей **SAP** и **SAR** и разместите их в отчете так:

```
— Report Summary Band ———————————————————————————————————————————
| Итого расход топлива:999.99  Перерасход (-), экономия (+): 999.99|
|_________________________________________________________________|
—————————————————————————————————————————————————————————————————
                        SAR                                SAP-SAR
```

Поля **SAP** и **SAR** создаются аналогично полям **SRP** и **SRR** с той разницей, что в позиции **Reset every** нужно выбрать опцию **<REPORT>**. Это значит, что запланированное и фактически израсходованное горючее всеми водителями, находящееся в полях **RGP** и **RGR**, будет суммироваться по перевозкам, сделан-

ным в октябре и выдаваться в конце отчета. Кроме того, будет вычисляться общий перерасход или экономия горючего. Перед созданием полей не забудьте установить курсор на нужную позицию в данной полосе отчета.

При создании поля **SAP** нужно учесть, что оно необходимо только для вычисления разности **SAP-SAR**, поэтому его не надо выводить. Поскольку это поле не будет выводиться, то при его создании курсор может стоять в любом место полосы. При создании поля **SAP**, как обычно, выберите позицию **Add field** меню **Fields** или нажмите клавишу **F5** и затем в высветившемся окне в колонке **SUMMARY** выберите позицию **Sum**, после чего в высветившемся окне установок параметров этого поля в позиции **Name** назначьте ему имя **SAP** (можно также в позиции **Description** ввести описание этого поля), а в позиции **Field to summarize on** - введите поле **RGP**, значения которого будут суммироваться и засылаться в поле **SAP**. Для того чтобы скрыть это поле, нужно в позиции **Hidden** установить значение **Yes**. После назначения всех параметров полю **SAP** не забудьте их сохранить клавишами **Ctrl-End**.

Поле разности **SAP-SAR** создается как и любое вычисляемое поле, например, аналогично полю **RGP** или полю разности **SRP-SRR**, разумеется, после предварительной установки курсора на ту позицию в полосе итогов, в которой оно должно размещаться. Для этого поля, как и для поля **SRP-SRR**, можно не назначать имени, в качестве которого по умолчанию будет служить само выражение.

Формирование структуры отчета закончено. Выбрав позицию **View report on screen** меню **Print**, можно просмотреть отчет на экране. Если ошибок в созданном документе нет, то выберите позицию **Save Changes and exit** меню **Exit**, введите имя файла отчета **MR1** в высветившемся окне **Save as** после имени диска и маршрута по директориям, например, **D:\DBASEIV\PRIMER\MR1**, и нажатием **Enter** сохраните программный файл отчета.

Запуск отчета на печать можно выполнить, не выходя из меню разработки отчета с помощью позиции **Begin printing** меню **Print** или из УЦ. В последнем случае нужно перед печатью отчета активизировать файл БД или запрос, по которому будут отбираться данные в отчет, затем установить курсор на имя отчета и нажать клавишу **Enter**. Высветится горизонтальное меню с позициями **Print report** и **Modify layout** и **Display data**. Выбор первой позиции высвечивает такое же меню, как и меню **Print** в форме разработки отчета. Позиция **Modify layout** высвечивает экран разработки отчета, а позиция **Display data** выводит на экран в стандартной форме **Browse** или **Edit** данные из связанного с отчетом файла БД или выборки.

Заметим, что отчет печатается кириллицей в том случае, если загружен соответствующий драйвер. Как уже упоминалось,

в меню **Print** предусмотрен вывод на несколько типов принтеров: **GENERIC, Ascii text, Epson FX-80**.

Драйвер с кириллицей обычно подключается к тому типу принтера, который задается при установке пакета dBASE IV. Поэтому, если кириллица не печатается, то надо в позиции **Printer model** после выбора подменю (позиции) **Destination** меню **Print** установить клавишами **Enter** или **Пробел** тот тип принтера, который был назначен при установке пакета, например, **Ascii text**. В этом же меню в позиции **Echo to screen** можно установить параллельный вывод отчета на принтер и экран. Полезной также является позиция **New page** подменю **Control of printer** меню **Print**, с помощью которой можно установить, также клавишами **Enter** или **Пробел**, перевод страницы перед печатью отчета (**BEFORE**), после печати отчета (**AFTER**), до и после печати отчета (**BOTH**) или вообще не переводить (**NONE**). В учебном примере установлено значение **NONE**. В этом же подменю с помощью задания значения **YES** в позиции **Wait between page** можно установить ожидание в работе принтера для заправки нового листа.

В подменю **Page dimensions** меню **Print**, выбрав позицию **Length of page**, можно задать количество строк на странице. Поскольку в учебном примере введено лишь несколько перевозок, которые не заполняют даже одну страницу, то число строк на печатаемой странице принято равным 33.

После выполненных назначений нажмите клавишу **Esc** и затем можно запустить отчет на печать, выбрав позицию **Begin printing** меню **Print**. Как упоминалось выше, все установки печати можно сохранить в специальном файле, выбрав позицию **Save settings to print form**. После этого высвечивается окно **Save as**, в которое нужно ввести имя файла установок. В качестве этого имени можно взять имя отчета **MR1**. Следует отметить, что перед запуском отчета на печать можно временно изменить для текущего сеанса печати все назначенные при создании отчета установки печати посредством выбора соответствующих позиций, но не сохраняя их в файле установок печати. Сохранение же изменений в файле установок означает замену старых значений на новые.

2.8. РАЗРАБОТКА МАРОК

Марки представляют собой небольшие формы, которые можно печатать по нескольку штук на листе. Они используются для печати почтовых открыток, товарных ярлыков и т.д. Если ширина марки кратна ширине листа, то они печатаются слева направо сверху вниз по листу бумаги. Если ширина марки равна ширине бумаги или меньше, но не кратна ей, марки будут печататься по вертикали. В каждой марке размещаются данные

одной записи файла БД. Максимальный размер марки - 255 строк и 255 колонок. Экранная область разработки форм марок аналогична области разработки отчетов в режиме размещения полей. Особенностью вывода на печать значений полей в марках по сравнению с отчетами является автоматическое удаление в них ведущих и конечных пробелов.

Меню разработки марок включает в основном те же компоненты, что и меню отчета. Отличие определяется спецификой формы марки. Ее структура задается меню **Dimensions** вместо меню **Bands** для отчета. Все остальные меню практически полностью совпадают с меню разработки отчета.

Меню **Dimensions** позволяет выбрать стандартные размеры марок или установить свои. Позиция **Predefined Size** предоставляет на выбор несколько стандартных размеров (по высоте и ширине) марок (в дюймах). Для тех размеров, которые дают возможность размещать несколько марок по ширине листа, приводится их максимальное число. Это число дается в соответствии с шагом печати символов шрифтом **PICA** (10 символов на дюйм).

Для установки собственных размеров используются позиции **Width of label** (ширина марки - числом символов) и **Height of label** (высота марки - числом строк). В форме марки с помощью позиции **Indentation** можно установить отступ абзаца при вводе текста. Расположение марок на бумаге устанавливается позициями **Lines between labels** (число пустых строк между рядами марок - до 16 линий) и **Spaces between label columns** (число пропускаемых позиций между марками в ряду - до 120 позиций). Количество марок в ряду определяется автоматически, исходя из заданных размеров, если не установлено их число с помощью позиции **Columns of labels** (до 15 марок в ряду).

Покажем действия по разработке марки на нашем примере. Предположим, что в **MBD1** помимо водителей введены данные на служащих предприятия. Требуется распечатать этикетки, приклеиваемые на рабочие шкафы служащих. Пример такой этикетки показан ниже.

```
┌──────────────────────────────────────┐
│         Ш К А Ф    N _____           │
│                                      │
│  Ответственный: Александров А.С.     │
└──────────────────────────────────────┘
```

Перед созданием марки откройте файл **MBD1**, затем установите курсор на позицию **<create>** окна **Labels** и нажмите **Enter**. Высветится экран создания (модификации) марок. Если Вы забыли открыть файл **MBD1**, содержащий фамилии, имена и отчества, то выберите позицию **Use different databases file or view** меню **Layout**, что позволит открыть его уже после выхода на экран разработки марки.

Затем надо определить размеры марки. Ширина марки устанавливается в опции **Width of label** меню **Dimensions** и, допустим, равняется 39, высота - 5 (позиция **Height of label** меню **Dimensions**), количество пустых строк между марками - 1 (**Lines between labels**), количество пробелов между колонками марок - 2 (**Spaces between labels columns**), число колонок - 2 (**Columns of labels**). При нажатии клавиши **Esc** окно марки на экране изменяет свои размеры в соответствии с параметрами, установленными в меню **Dimensions**.

Теперь установите курсор в нужную позицию и введите текст "ШКАФ N_ _ _" и "Ответственный:".

Следует отметить, что в неадаптированной версии dBASE IV при разработке экранных форм, отчетов, марок и прикладных программ не вводятся русские прописные буквы "**К**" и "**Н**", которые нужно заменять соответствующими латинскими буквами. Затем нужно ввести в марку поля, которые будут выводиться, - **FAM**, **NAM** и **OTC**. Для этого используется позиция **Add field** меню **Fields**, а из высветившегося списка полей файла БД - выбирается требуемое поле. Затем высветится окно с установками параметров поля, которые при вводе поля **FAM** изменять не будем и сохраним эти установки клавишами **Ctrl-End**, после чего поле будет размещено в текущей позиции курсора. Не забудьте перед размещением каждого поля устанавливать курсор на требуемую позицию.

При вводе поля **NAM** нужно изменить шаблон, чтобы высвечивалась только первая буква и точка после нее. Для этого выберите позицию **Template** в окне с параметрами поля и измените шаблон поля, оставив только один символ **X** с точкой (**"X."**). Нажмите **Enter** для фиксации изменения и затем **Ctrl-End** для сохранения параметров поля. Аналогично введите в марку поле **OTC**. В итоге шаблоны полей **FAM**, **NAM** и **OTC** будут выглядеть следующим образом: **"XXXXXXXXXXXXXXX X.X."**.

Для обведения этикетки рамкой переместите курсор в левый угол марки, нажмите клавишу **Alt** и не, отпуская ее, нажмите последовательно на правой цифровой клавиатуре цифры **2,1,8** (код ASCII символа левого верхнего уголка). После этого отпустите клавишу **Alt**, и на экране высветится левый верхний уголок. Обозначим нажатие этих клавиш как **Alt-218**. В правом верхнем, правом нижнем и левом нижнем углах марки нажмите соответственно клавиши **Alt-191**, **Alt-217** и **Alt-192**. Теперь остается только соединить уголки горизонтальными и вертикальными линиями, нажимая нужное число раз соответственно клавиши **Alt-196** и **Alt-179**. Этикетка на шкаф готова. Для проверки просмотрите созданную форму этикеток на экране, выбрав позицию **View labels on screen** меню **Print**.

Наконец, нужно установить опции печати. При условии, что этикетки будут распечатываться на стандартном листе в две колонки, опции будут следующие: количество строк на странице

(позиция **Length of page** подменю **Page Dimensions**) - 60, отступ слева (позиция **Offset from left** этого же подменю) - 2. Для установки ожидания перед печатью очередного листа установите опцию "Ждать между страницами" (позиция **Wait between pages - YES** подменю **Control of Printer**), а также опцию "Вывод на принтер" (позиция **Write to printer** подменю **Destination**).

Установки печати целесообразно сохранить в специальном файле, который будет автоматически вызываться при печати марки. Запись установок в этот файл осуществляется выбором позиции **Save settings to print form**. После этого высвечивается окно **Enter name of print file** с текущим маршрутом по директориям. Введите имя, например, **MUST**, которое нужно зафиксировать клавишей **Enter**.

Для сохранения программного файла марки на диске выберите позицию **Save this label design** меню **Layout** и введите имя разработанной марки - **ML1** в окно **Save as** после текущего диска и директория. Далее нажмите клавишу **Enter**. Запуск марки на печать без выхода из экранной формы можно выполнить, выбрав позицию **Begin printing** меню **Print**.

Выход в УЦ из экрана создания (модификации) марки осуществляется выбором любой из позиций меню **Exit**, поскольку ее файл уже записан на диск. Для вывода марки на печать из УЦ нужно выбрать ее имя в окне **Labels**, затем в высветившемся меню выбрать позицию **Print label**, после чего откроется такое же меню, как и при выборе меню **Print** в случае запуска марки из экранной формы ее разработки, в котором можно выбрать опцию для высвечивания марки на экране (**View labels on screen**) или начать печатать (**Begin printing**). Перед этим можно изменить ранее сделанные установки печати (позиции **Output options**, **Page dimensions** и др.).

2.9. ГЕНЕРАТОР ПРИКЛАДНЫХ ПРОГРАММ

Генератор прикладных программ позволяет создавать программные файлы (программы) на языке dBASE IV пользователям, не знающим команд этого языка. Эти программы дают возможность работать с ранее созданными файлами БД, экранными формами ввода и вывода информации, запросами на выборку и обновление данных в файлах БД, отчетами и марками. Кроме того, в программах генерируются меню выполнения операций с указанными выше объектами. В прикладную программу можно также вводить команды языка dBASE IV, включая разработанные ранее процедуры и программные модули. Для подготовленных пользователей и программистов генератор дает возможность уменьшить трудозатраты на разработку сложных программ посредством использования заранее разработанных унифицированных модулей.

Собственно генератор прикладных программ позволяет создавать следующие компоненты: главную процедуру программы, горизонтальные и вертикальные меню и списки файлов, полей файлов БД/выборок, значений поля файла БД/выборки. Списки похожи на меню - курсор может передвигаться по позициям списка, а после выбора позиции (клавишей **Enter**) выполняется заданное действие. Отличие меню от списка состоит в том, что каждой позиции меню можно назначить свое действие, а списку только одно общее действие. Ниже для общности под термином меню будут пониматься также и списки, а при упоминании файла БД под ним будет также подразумеваться и выборка. Кроме этих компонентов генератор позволяет создать особый компонент - программный модуль **Batch**, который может назначаться для выполнения при выборе позиций меню или списков. Можно также назначить его запуск самостоятельно, например, в начале выполнения прикладной программы. Этот модуль создается как меню вертикального типа. Каждой позиции меню модуля **Batch** назначается определенное действие. Текст позиции служит для пояснения выполняемого действия, как и в обычном меню. Отличие модуля **Batch** от меню состоит в том, что меню модуля **Batch** при его вызове не высвечивается на экране и все действия, назначенные позициям его меню, выполняются в пакетном режиме - как один модуль. По существу, модуль **Batch** предназначен для неподготовленных пользователей как инструмент по созданию обычных программных модулей.

Меню прикладной программы можно реализовать в виде сложной многоуровневой системы, включающей меню горизонтального и вертикального типов и списки. Меню состоит из главного меню и прикрепленных к его позициям меню нижеследующих уровней. В качестве главного меню может быть меню горизонтального или вертикального типов. Прикладная программа может также начинаться выполнением модуля **Batch**. Позициям меню назначаются операции с файлами и другие действия.

Перед и после вызова меню или выполнения позиций меню можно назначить выполнение команд dBASE IV, например, команду вызова процедуры, написанной на языке dBASE IV или SQL. В состав программы, создаваемой генератором, входит главная процедура, в которой делаются установки параметров и опций SET-команд, переменных и т.д. и вызываемые из главной процедуры программные модули и команды, а также меню и процедуры. С помощью меню, посредством выполнения процедур или команд можно вызывать заранее разработанные запросы на обработку файлов БД, экранные формы ввода и вывода информации, отчеты, марки, а также различные процедуры и функции. Использование этих возможностей УЦ и генератора прикладных программ в сочетании с программированием дополнительных объектов на языке dBASE IV и SQL

позволяет создавать прикладные программы любой степени сложности.

Разработка прикладной программы выполняется с помощью меню генератора, включающего следующие позиции: **Designe**, **Application**, **Generate**, **Preset**, **Exit**. При разработке меню, списка или модуля **Batch** вместо позиции **Application**, служащей для разработки главной процедуры, высвечиваются соответственно позиции **Menu**, **List** или **Batch** и добавляется общая для всех позиция **Item**. Собственно разработка компонентов прикладной программы выполняется с помощью позиций (меню) **Designe**, **Application**, **Menu**, **List**, **Batch** и **Item**. Остальные позиции выполняют служебные функции: **Generate** - генерацию прикладной программы, **Preset** - предварительную установку параметров dBASE IV, **Exit** - выход из генератора прикладных программ.

Перед разработкой прикладной программы целесообразно назначить общие установки и параметры dBASE IV, которые распространяются на всю прикладную программу. Особенно это полезно, если предполагается разрабатывать серию прикладных программ, в которых используются унифицированные компоненты. Меню **Preset** содержит опции, позволяющие назначить текущий диск и маршрут по директориям для файлов, которые будут обрабатываться в прикладной программе, данные об авторе прикладной программы, авторских правах и используемой версии dBASE IV (могут высвечиваться в заставке при запуске прикладной программы), а также некоторые установки dBASE IV, включая цвет различных областей экрана, подачу звукового сигнала при определенных ситуациях, ряд установок для обработки файлов БД и др. В дальнейшем установки, сделанные в меню **Preset**, можно изменить, дополнить и конкретизировать при разработке компонентов прикладной программы в соответствии с их функциями.

Для назначения диска и маршрута по директориям выбирается опция **Application drive/path** меню **Preset**, после чего в соответствующие графы высветившегося шаблона вводится имя диска (**Drive**) и маршрут по директориям (**Search path**). Опция **Sign-on defaults** служит для ввода данных в заставку прикладной программы. После выбора этой опции высвечивается шаблон, в графы которого требуется ввести данные: имя автора (**Author**), авторские права (**Copyright notice**) и версию используемого пакета dBASE IV (**dBASE version**). Следует отметить, что эти данные будут высвечены в заставке не текущей, а только следующей разрабатываемой прикладной программой. Сохранение введенных в шаблоны значений выполняется клавишами **Ctrl-End**. Ниже при описании различных меню всегда будет подразумеваться, что после ввода данных в шаблон их нужно сохранить клавишами **Ctrl-End**.

Опция **Environment setting** позволяет включить (**ON**) или выключить (**OFF**) клавишей **Пробел** установки на выполнение

следующих действий: подачу звукового сигнала при определенных ситуациях, например, после заполнения поля файла БД данными (**Set BELL**), копирование содержимого поля из предыдущей записи в последующую (**Set CARRY**), вывод четырех цифр года (включая цифры столетия) в дате (**Set CENTURY**), автоматический переход в следующее поле записи после заполнения предыдущего (**Set CONFIRM**), прерывание выполнения задачи клавишей **Escape** (**Set ESCAPE**), высвечивание предупреждающего сообщения при удалении файла БД (**Set SAFETY**) и установку нестандартного разделителя полей файлов БД (**Set DELIMETERS to**). При включении звукового сигнала можно установить его частоту и длительность.

Цвета экрана устанавливаются опцией **Display options** посредством выбора позиций: цвета текста и заголовков (**Standard - All**), цвета внутренней части рамок компонентов, поля и подсвеченные позиции (**Enhanced - All**), цвет текста, а также не активные поля на экране в форме **Browse** (**Normal text**), цвет сообщений в нижней части экрана, а также не активные меню и списки (**Messages**), цвет заголовков типа имен полей в таблице Browse, а также линий в этой таблице (**Titles**), цвет подсвеченных позиций меню и списков (**Highlight**), цвет внутренней части рамок меню и списков (**Boxes**), цвет информационной строки, часов, рамки сообщений об ошибках (**Information**), цвет активных полей файлов БД в таблице **Browse** и других редактируемых полей (**Fields**).

Разработка прикладной программы обычно начинается с создания главной процедуры. Экран разработки главной процедуры появляется после выбора позиции **<create>** в окне **Applications** на панели УЦ и затем опции **Modify application** и представляет собой шаблон с графами, в которые требуется ввести следующие данные: имя главной процедуры (**Application name**), тип главного меню программы (**Main menu type**), имя этого меню (**Main menu name**), файл БД, используемый в программе (**Database/View**), индексные файлы типа **.ndx** или **.mdx**, связанные с этим файлом БД (**Set index to**), главный (управляющий) индекс типа **.ndx** или указатель множественного индексного файла типа **.mdx** (**ORDER**) и словесное описание главной процедуры (**Description**). Управляющий индекс (указатель индекса) можно задать в виде его имени или порядкового номера из списка файлов типа **.ndx** или указателей файлов типа **.mdx**.

Выбор типа главного меню выполняется клавишей **Пробел** из трех значений (**Bar** - горизонтальное, **Pop-up** - вертикальное меню, модуль **Batch**). Имя файла БД можно ввести с клавиатуры или выбрать из списков файлов БД в текущем и других директориях дисков, высвечиваемых клавишами **Shift-F1**.

В верхней части этого списка имеется позиция **<create>**, позволяющая создать новый файл БД, ввести в него данные и

затем вернуться в генератор прикладных программ в ту точку, из которой был выполнен выход на создание файла БД. Эту же операцию можно выполнить для всех типов файлов - индексных файлов, запросов, экранных форм, отчетов и марок, файлов меню, списков и модуля **Batch**. В дальнейшем, когда будет упоминаться ввод имени любого типа из перечисленных файлов или имен полей файла БД, всегда будет предполагаться возможность ввода имени с клавиатуры или выбора этого имени из списков в директориях дисков, высвечиваемых клавишами **Shift-F1**, а также возможность их немедленного создания с помощью позиции **<create>** этого списка.

После сохранения введенных в шаблон главной процедуры данных высвечивается окно, представляющее собой заставку, которая будет появляться при запуске созданной прикладной программы. Внутри окна можно записать свой текст, а также изменить размеры окна и переместить его на другое место экрана. Изменение размеров окна выполняется навигационными клавишами после нажатия клавиш **Shift-F7**. Новый размер окна фиксируется клавишей **Enter**. Для передвижения окна сначала нажимается клавиша **F7**, затем окно можно передвигать навигационными клавишами. Новое место окна также фиксируется клавишей **Enter**.

Дальнейшие действия по созданию главной процедуры выполняются с помощью меню **Application**. Опция **Name and describe** этого меню позволяет подтвердить или изменить введенное ранее имя и словесное описание главной процедуры, а опция **Assign main menu** - подтвердить или изменить имя ранее введенного главного меню. Сохранение введенных значений выполняется клавишами **Ctrl-End**. Для подтверждения ввода в прикладную программу заставки выбирается опция **Display sign-on banner** (ответ **Yes** на предложение **Display frame as "sign on banner" at run time**). Если ответ **No**, то при запуске прикладной программы сразу высвечивается главное меню. Опция **Edit program header comments** осуществляет ввод в комментарий текста сгенерированной программы данных об авторе прикладной программы (**Application author**), сведений о правах на копирование (**Copyright notice**) и используемую версию dBASE IV (**dBASE version**). После генерации программу с комментарием можно вывести на печать в режиме ввода команд с клавиатуры, используя команду **Type <имя программы> on printer** или с помощью меню **Print** текстового редактора dBASE IV.

Опция **Modify application environment** дает возможность изменить назначение текущего диска, маршрут по директориям, ввести имя файла БД, который будет обрабатываться в прикладной программе, и связанные с ним индексные файлы, а также изменить другие установки dBASE IV, ранее выполненные в меню **Preset**. После выбора этой опции высвечивается меню,

позволяющее ввести необходимые данные. Три опции этого меню (**Display options**, **Environment setting**, **Search path**) полностью совпадают с опциями меню **Preset**. Четвертая опция **View/database and index** используется для ввода имени файла БД, связанного с ним индексного файла и указателя индекса в множественном индексном файле. После выбора этой опции в высветившийся шаблон в соответствующие графы вводится (если не введено раньше) имя файла БД (**Database/view**), имя одного или нескольких индексных файлов типа **.ndx** или **.mdx** (**Set index to**), а также имя главного (управляющего) индекса (указателя индекса для файла типа **.mdx**) или его порядковый номер (**Order**).

Опция **Generate quick application** меню **Application** позволяет создать стандартную прикладную программу с одним вертикальным меню, в котором автоматически формируются позиции (на английском языке) для добавления, изменения, вывода на экран в стандартной форме **Browse** и удаления записей файла БД. Генератор автоматически присваивает этой программе имя **Quckapp**. Текст позиций меню можно после генерации программы заменить на русский язык при модификации программы. Для этой программы можно также задать поиск по индексу, выдачу на печать ранее сформированного отчета или марки, а также ввод и вывод информации в ранее разработанной экранной форме. После выбора этой опции нужно ввести (или изменить) в соответствующие графы шаблона следующие данные: имя файла БД, предназначенного для обработки в программе (**Database file**), имя экранной формы (**Screen format file**), имя отчета (**Report format file**), имя марки (**Label format file**), имя одного или нескольких индексных файлов (**Set INDEX to**), имя или число, определяющее порядковый номер главного индекса (указателя индекса) в списке индексных файлов типа **.ndx** или файла типа **.mdx**. (**Order**), имя автора прикладной программы, которое вводится в качестве комментария в программу и документацию, создаваемую генератором прикладных программ (**Application author**), заголовок меню, создаваемого генератором (**Application menu heading**).

Сохранение всей введенной информации выполняется клавишами **Ctrl-End**, после чего высвечивается предложение на генерацию программы (**YES** - программа генерируется, **NO** - нет). Если программа не создана, то ее можно сгенерировать впоследствии, выбрав позицию **Select template** меню **Generate** и затем введя **Quickapp.gen** - имя файла шаблона этой прикладной программы, используемого при генерации.

Для сохранения разработанной главной процедуры нужно выбрать позицию **Save current application definition** меню **Application**, которая записывает на диск файл разработки главной процедуры с расширением **.app**. В этом меню еще имеется позиция **Clear work surface**, которая используется обычно при

разработке других компонентов прикладной программы - меню, списков и модулей **Batch** - и позволяет удалить с рабочей поверхности экрана все компоненты, кроме главной процедуры (на экране главная процедура высвечивается в виде рамки заставки).

Следующим этапом разработки прикладной программы обычно является создание меню. Для этого используется меню **Designe**, позволяющее создавать горизонтальные, вертикальные меню, модуль **Batch** и списки. Эти компоненты записываются на диск в виде файлов с различными расширениями, идентифицирующими их тип: **горизонтальные (.bar)** и **вертикальные (.pop) меню, модуль Batch (.bch), списки файлов (.fil), списки полей файлов БД (.str), списки значений полей файлов БД (.val).** Создание определенного типа меню, списка и модуля **Batch** осуществляется выбором опций меню **Designe:** горизонтальное меню (**Horizontal bar menu**), вертикальное меню (**Pop-up menu**), список файлов (**Files list**), список полей файла БД (**Structure list**), список значений поля файла БД (**Values list**), модуль **Batch** (**Batch process**).

После выбора опции создания горизонтального меню высвечивается список, из которого можно выбрать существующие в текущем директории файлы этого типа меню. Для создания нового горизонтального меню требуется выбрать позицию **<create>** в этом списке. Затем высвечивается окно, в графы которого требуется ввести: имя меню (**Name**), словесное описание меню (**Description**), текст сообщения, который будет высвечиваться внизу экрана при выборе данного меню в ходе выполнения прикладной программы (**Message line prompt**). Введенные данные следует сохранить клавишами **Ctrl-End**. После этого в верхней части экрана высвечивается рамка горизонтального меню и курсор перемещается в левую часть этой рамки, в которую теперь можно вводить текст позиций меню. Текст позиции вводится после нажатия клавиши **F5**. Завершается ввод текста этой же клавишей. Затем нужно курсор передвинуть вправо по рамке на требуемое место и аналогично ввести текст следующей позиции.

При создании остальных компонентов (вертикального меню, всех типов списков и модуля **Batch**) выполняются те же операции вплоть до высвечивания рамок этих компонентов. Изменение размеров рамок (кроме списков) и их перемещение выполняется так же, как это описано выше для рамки заставки прикладной программы. Текст в позиции вертикального меню и модуля **Batch** вводится с клавиатуры, а переход по позициям осуществляется навигационными клавишами **Вверх** и **Вниз**. В позиции списков текст не вводится.

Если ранее уже были созданы меню или списки, их можно выбрать из перечня, высвечиваемого при выборе соответствующих позиций меню **Designe**. После того, как созданы все

необходимые компоненты, можно приступить к назначению позициям меню и спискам требуемых действий или назначению выполнения модуля **Batch**. Установленное ранее в главной процедуре назначение конкретного файла БД, параметров и других значений dBASE IV распространяется также и на меню. Однако эти назначения можно изменить для каждого компонента прикладной программы, используя их меню: **Menu** - для меню горизонтального и вертикального типа, **Batch** - для модуля **Batch**, **List** - для списков. Поэтому перед присвоением позициям меню действий может потребоваться обращение к этим меню.

В указанных меню ряд опций (по назначению параметров dBASE IV, изменению имени компонента и др.) полностью аналогичен опциям меню **Application** и описываться не будут. Другие опции являются общими для всех компонентов, а некоторые специфичны для определенного компонента. Рассмотрим сначала опции, общие для всех компонентов. К ним относятся опции, служащие для изменения файла БД, предназначенного для обработки в компоненте, записи справочного текста, который можно высветить клавишей **F1** в активном состоянии компонента, ввода команд dBASE IV, которые будут выполняться до или после активизации компонента на экране и др.

Опция **Override assigned database or view** дает возможность подтвердить или изменить ранее назначенный файл БД или использовать тот файл БД, который будет активным в момент работы компонента. Сначала высвечивается имя файла БД и индексных файлов, ранее назначенное компоненту. Для изменения этих назначений заполняются графы в высвечиваемом шаблоне. В графе **For this <компонент> you may use values** следует выбрать клавишей **Пробел** одно из трех ключевых слов: **IN EFFECT AT RUN TIME, ABOVE, ENTERED BELOW**.

Первое ключевое слово позволяет назначить для обработки файл БД, который будет активным в момент работы компонента. Например, при выполнении прикладной программы сначала может высвечиваться список файлов БД, из которого пользователь выбирает требуемый файл, а затем появляется меню операций с этим файлом. Второе ключевое слово фиксирует назначение файла БД в главной процедуре. В этих двух случаях последующие графы в шаблоне на экране не заполняются. При выборе третьего ключевого слова графы шаблона **Database/view, Set INDEX to, ORDER** заполняются так же, как и в главной процедуре.

Опция **Write help text** (отсутствует в меню **Application** и **Batch**) служит для записи справочного текста, который высвечивается клавишей **F1** в активном состоянии компонента на экране в ходе выполнения прикладной программы. Справочный текст записывается с клавиатуры на полном экране (в объеме до **19** строк). В тексте запрещаются кавычки. После записи текст следует сохранить клавишами **Ctrl-End**. В этой опции справочный

текст можно ввести для компонента в целом, а для каждой позиции горизонтального и вертикального меню он вводится с помощью позиции **Item** основного меню генератора прикладных программ. В последнем случае справочный текст для компонента в целом не высвечивается.

Опция **Modify display options** (отсутствует в меню **Batch** и **Application**, где аналогичные действия выполняются опцией **Modify application environment**) позволяет установить тип рамки и цвета элементов компонента на экране. Эти же установки переносятся на компоненты, которые разработаны вне генератора прикладных программ и вызываются из меню, например, экранные формы ввода и вывода информации, отчеты и т.д.

Эти назначения перекрывают установки, выполненные в меню **Preset** генератора прикладных программ, а также в опции **Display options** меню **Application**. Тип рамки включает: одиночную (**Single**) и двойную линии (**Double**), сплошную полосу (**Panel**) и без рамки (**None**).

Опция **Embed code** позволяет ввести команды языка dBASE IV, которые выполняются в составе прикладной программы перед или после активизации компонента. Для этого выбираются опции **Before** (перед) или **After** (после), после чего открывается полный экран для ввода команд (до 19 строк). Для ввода большего количества команд можно предварительно создать процедурный модуль любой длины с расширение .prg и вызвать его из прикладной программы, введя команду **DO <имя процедуры>.prg.**

К числу опций, специфичных для определенного типа меню, относится опция **Attach pull-down menus** меню **Menu**, позволяющая прикрепить к позиции горизонтального меню вертикальное меню или один из списков. После выбора данной опции высвечивается сообщение **Pull down associated menus...** Если ответ **Yes**, меню или список прикрепляется, **No** - нет. Прикрепляемое меню или список имеют те же атрибуты, что и горизонтальное меню. Следует отметить, что в данной опции устанавливается только возможность прикрепления меню или списка. Собственно прикрепление выполняется опцией **Open a menu** позиции **Item** основного меню генератора прикладных программ.

В меню **List** имеются три опции, отражающие специфику данного меню. Опция **Identify files in list** предназначена для ввода из текущего директория имен файлов, которые должны быть в списке. Для этого используется шаблон имени файла со знаками замещения ***** и ?. Например, ***.dbf** позволяет ввести в список из директория только файлы типа **.dbf**, а шаблон ***.?dx** - индексные файлы типа **.ndx** и **.mdx**.

Опция **Identify fields in list** предназначена для создания списка полей файла. После выбора этой опции следует ввести с клавиатуры имена полей, которые требуется включить в список или выбрать их из списка полей файла, высвечиваемого клавиша-

ми **Shift-F1**. При вводе с клавиатуры имена полей должны разделяться запятыми. Если не введено ни одно имя поля, то в список включаются все поля текущего файла БД. Если создаваемый список полей прикрепляется к горизонтальному меню, то в него включаются все поля файла БД.

Опция **Identify field values in list** предназначена для создания списка значений поля. После выбора этой опции высвечивается предложение ввести имя поля, из которого составляется список значений. Имя поля вводится с клавиатуры или выбирается из списка полей текущего файла БД, высвечиваемого клавишами **Shift-F1**. Сохранение выбранных полей выполняется клавишами **Ctrl-End**.

Для сохранения на диске активного компонента (в нем находится курсор, и его рамка подсвечивается другим цветом) выбирается опция **Save current** (после сохранения компонент остается на экране, что позволяет продолжать с ним работу), для удаления с экрана с сохранением - опция **Save changes** и для удаления без сохранения - опция **Abandon changes** позиции **Put away current**.

В процессе разработки на экране может находиться несколько компонентов прикладной программы (их общее количество определяется размером свободной оперативной памяти ПК), но активным может быть только один компонент. Передвижение курсора по компонентам (с одновременным переводом его в активное состояние) выполняется клавишами **F3** и **F4**.

Если требуется удалить с экрана сразу все компоненты (главная процедура не удаляется), используется опция **Clear work surface**, которая удаляет последовательно все компоненты, высвечивая перед удалением меню с позициями: **Save changes** - сохранить изменения, **Abandon changes** - без сохранения изменений.

Меню **Item** позволяет назначить позициям горизонтального, вертикального меню, модуля **Batch** и спискам основное и дополнительные действия, которые будут выполняться при выборе позиций и списков. К дополнительным действиям относится, например, вывод клавишей **F1** на экран справочного текста, поясняющего основное действие позиции, или установка указателя записи файла БД перед выполнением основного действия.

Перед назначением действий нужно установить курсор на соответствующую позицию меню или любую позицию списка (напомним, что всем позициям списка может назначаться только одно действие). Дальше для сокращения при описании назначений действий позициям меню под ними будут также подразумеваться и списки.

Опции позиции **Item** основного меню генератора прикладных программ выполняют следующие функции: назначение позиции

меню основного действия, назначение дополнительного действия, ввод команд dBASE IV до или после выполнения основного действия, выполняемого выбором позиции меню, изменение назначения файла БД, который будет обрабатываться при выборе данной позиции меню, и вывод на экран данных о текущей позиции меню.

Для назначения основного действия используется опция **Change action**. После выбора этой опции высвечивается меню с описанием основных действий. В их число входят: вывод на экран записей файла БД в форме **Browse** и **Edit**, вывод на принтер или экран данных в форме отчетов или марок, выполнение операции с файлами БД (копирование, добавление записей из другого файла, копирование отдельных полей, замена содержимого полей, пометка записей на удаления, уничтожение пометок на удаление, удаление из файла помеченных записей, создание индексного файла, переиндексирование файла БД, сортировка записей с созданием нового файла, импорт внешних и экспорт своих файлов), выполнение программ, процедур и групп команд dBASE IV, DOS, двоичных модулей и макроопределений, назначение выполнения модуля **Batch**, не выполнение никакого действия (назначается по умолчанию), выход в УЦ или DOS.

После выбора опции **Change action** высвечивается меню с перечнем отдельных операций и групп операций. Назначение выполняется выбором требуемой операции. Операция **Browse** выводит на экран данные из файла в табличной форме. После выбора этой опции высвечивается шаблон с графами, определяющими структуру и форму таблицы **Browse**. В графу **FIELDS** вводятся имена полей для включения в экранную форму **Browse**. Имена полей можно набрать с клавиатуры, разделяя их запятыми, или выбрать из списка, высвечиваемого клавишами **Shift-F1**. Область для ввода имен полей имеет длину 68 символов. Если имена полей не вводятся в шаблон, то в экранной форме **Browse** высвечиваются все поля. Выбор имен полей из списка выполняется клавишей **Enter**.

В графу **FILTER** можно ввести запрос и (или) фильтр для отбора требуемых записей из файла БД. При вводе имени запроса перед ним следует поместить ключевое слово **FILE**. В таблице **Browse** с большим количеством полей можно установить число смежных полей, начиная с первого, которые фиксируются слева на экране при движении курсора по колонкам таблицы. Это число указывается в графе **Fields to LOCK on screen** (0 означает, что нет фиксированных полей). Иногда, при обработке полей записи файла БД, желательно во избежание порчи данных установить защиту от обновления для некоторых полей. Графа **FREEZE edit for field** позволяет указать поле, которое можно обновлять, а для остальных полей устанавливается запрет на редактирование. Если в эту графу не введено ни одного имени

поля, то будет разрешено редактирование всех полей (в том числе ранее защищенных от редактирования). Кроме этого, можно также установить еще несколько режимов защиты информации по отношению ко всему файлу в целом. Установка в графу **Allow record ADD** клавишей **Пробел** значения **Yes** разрешает и **No** запрещает добавление записей в конец файла БД. Аналогично в графе **Allow record EDIT** устанавливается разрешение или запрет обновления, а в графе **Allow record DELETE** то же для удаления записей в файле БД.

Для удобства работы с данными в полях файла БД в таблице **Browse** можно установить в графе **Maximum column WIDTH** требуемую ширину высвечиваемой колонки каждого поля независимо от ширины, принятой при создании файла БД.

Ввод и вывод информации из файлов БД можно выполнять не только в стандартной форме **Browse**, но и в экранных формах, разработанных пользователем. Такую ранее разработанную форму можно использовать вместо стандартной формы **Edit**, которая является альтернативой формы **Browse** (переключение осуществляется клавишей **F2**). В графу **FORMAT file** вводится имя экранной формы (с расширением **.fmt**), которая будет высвечиваться вместо стандартной формы **Edit**. Имя экранной формы вводится с клавиатуры или выбирается из списка, высвечиваемого клавишами **Shift-F1**. Можно также создать новую экранную форму, выбрав позицию **<create>** этого списка. Можно также использовать ранее использовавшуюся структуру стандартной экранной формы **Browse**. Для этого служит графа **Use PREVIOUS Browse table**. Значение **Yes** назначает ранее использовавшуюся экранную форму **Browse**, а **No** позволяет создавать новую. Графа **FOLLOW record after update** устанавливает режим перестановки записи в файле БД в соответствии с его новым значением индекса (**Yes**) после ее обновления или режим без перестановки обновленной записи (**No**). В нескольких графах шаблона работы с таблицей **Browse** можно установить опции, определяющие внешний вид этой экранной формы. В графе **Display Browse MENU** можно удалить (**No**) или оставить (**Yes**) меню экранной формы **Browse**, в графе **COMPRESS display** - сжать заголовок экранной формы (**Yes**), что позволяет высвечивать в таблице **Browse** на 2 записи больше, чем обычно, в графе **KEEP image on exit** - оставить (**Yes**) или удалить (**No**) экранную форму после завершения сеанса операций с файлом БД. Здесь и ниже выбор альтернативных значений граф шаблонов осуществляется клавишей **Пробел**.

Операция **Edit** выводит на экран по одной записи файла БД в альтернативной стандартной форме, в которой поля расположены по вертикали. При назначении этой операции можно также задать с помощью шаблона ряд вариантов (опций) ее использования. Эти опции в основном совпадают с опциями формы **Browse** с небольшими отличиями. В графе **Mode** клавишей

Пробел устанавливается общий режим работы с формой **Edit**: обновление - **Edit** и добавление - **APPEND**. В последнем случае задействуются только опции **Format** и **Fields**. В режиме **Edit** можно также задать условия отбора записей: в графе **SCOPE** - диапазон обрабатываемых записей файла БД, в графе **FOR** - условие отбора записей, в графе **WHILE** - условие, определяющее конец обработки записей. Диапазон обрабатываемых записей устанавливается ключевыми словами: **RECORD <N>** - одна запись с номером N, **NEXT <N>** - N записей, начиная с текущей, **ALL** - все записи и **REST** - записи, начиная с текущей до конца файла.

Опция **Display or print** меню **Change action** позволяет назначить операцию вывода отчета или марки, установить внешнее устройство этого вывода, а также некоторые опции вывода на экран файла БД. Для назначения вывода на печать выбирается опция **Report** меню, высвечиваемого после выбора опции **Display**. После этого высвечивается шаблон, в который вводятся данные, необходимые для вывода отчета. В графу **FORM name** вводится имя ранее разработанного отчета. Обычно сформированный отчет уже включает все элементы для его вывода, в том числе условия для отбора записей, структуру заголовка страницы и т.д. Если же в отчет не включено условие отбора записей, то его можно ввести из разрабатываемой прикладной программы. Для этого служат графы шаблона **FILTER**, **SCOPE**, **FOR** и **WHILE**, которые заполняются так же, как и для экранных форм **Browse** и **Edit**. В графе **HEADING** можно задать ввод дополнительного заголовка, который будет печататься в первой строке каждой страницы отчета (в этом заголовке нельзя использовать двойные кавычки), а в графе **Heading format** - вывод в заголовке даты и номера страницы. Для этого клавишей **Пробел** выбирается значение **INCLUDE DATE AND PAGE** (при выборе **PLAIN** дата и номер страницы не выводятся).

Кроме этого, в графе **Report format** можно задать (также клавишей **Пробел**) вывод на печать всех записей (**FULL DETAIL**) или только строк, содержащих итоги и подытоги (**SUMMARY ONLY**). В графе **Before printing** можно установить этой же клавишей перевод страницы перед печатью (**SKIP TO NEW PAGE**) или печать без предварительного перевода страницы (**DO NOT EJECT**). В графе **Send output to** устанавливается внешнее устройство, на которое должен выводиться отчет: принтер (**PRINTER**), файл на диске (**DISK FILE**), экран (**SCREEN**), или по запросу пользователя в ходе выполнения программы (**ASK AT RUN TIME**). В последнем случае в ходе выполнения программы высвечивается вертикальное меню с тремя позициями, указывающими вывод на принтер, диск или экран.

Опция **Labels** позволяет назначить позиции меню вывод марки. После выбора этой опции высвечивается шаблон с графами, в которые требуется ввести необходимые данные.

В шаблоне марки графы в основном те же, что и в шаблоне отчета, хотя несколько граф, специфичных для отчета, отсутствуют. Одна графа - **Print SAMPLE** - специфична только для марки, в ней устанавливается пробный вывод марки (значение **Yes**) перед печатью.

Опция **Display/list** определяет некоторые параметры вывода на экран записей файла БД. Эти параметры также требуется ввести в графы высвечиваемого после выбора опции шаблона. В их число входят выводимые поля (**Fields**), условия отбора записей файла БД (**FILTER, SCOPE, FOR** и **WHILE**) и внешнее устройство вывода (**Send output to**). Эти графы заполняются так же, как и в шаблонах экранных форм (операций) **Browse, Edit** и др. Кроме них включены графа **Pause at full page/screen** для установки паузы после вывода 20 строк (значение **Yes**) или вывода без перерыва (**No**) и графа **Include RECORD NUMBERS** для установки вывода номера записи (значение **Yes**).

Опция **Perform file operation** меню **Change action** дает возможность назначить позиции меню или модуля **Batch** прикладной программы одну из операций с файлами БД. После выбора этой опции высвечивается меню, позволяющее выбрать конкретную операцию. Операция копирования файлов назначается выбором опции **File copy**, после чего в высветившийся шаблон следует занести имена файлов источника (**Copy file**) и мишени (**to file**).

Для добавления записей из заданного файла (или из массива переменных) в активный файл БД в момент выполнения прикладной программы (опция **Add records from file**) требуется ввести имя файла (массива переменных) - источника в графу шаблона **Add records FROM file/array**, а в графе **type** выбрать одно из ключевых слов, определяющих тип файла или массив (включая внешние файлы): **DBF, dBASE II, FW2, RPD, DELIMITED, SDF, SYLK, WKS, ARRAY** (ключевое слово массива переменных). Если выбирается тип **DELIMITED**, то требуется ввести знак разделителя (по умолчанию принимаются двойные кавычки). В графу **FOR** вводится условие, определяющее отбираемые записи для добавления в файл БД. Копирование заданных полей активного файла в момент выполнения прикладной программы в другой файл (или в массив переменных) назначается выбором опции **Copy records to file**. Для этого в графы высветившегося шаблона вводятся данные, аналогичные операции добавления: имя файла БД или массива переменных (**Copy records TO file/array**), тип файла, а также условия отбора копируемых полей и записей (**FIELDS, FILTER, SCOPE, FOR** и **WHILE**).

Замена содержимого полей файла БД осуществляется выбором опции **Substitute field values**. При этом в графы шаблона следует ввести имена полей, в которых будет заменяться содержимое (**The field named**), и новые значения этих полей

(**with this value**). В графе **ADDITIVE** устанавливается режим изменения содержимого поля (только для **Memo** поля): замещение (значение **No**) или добавление к уже имеющемуся содержимому поля (**Yes**). Кроме этого, можно указать условия отбора записей, для которых выполняется операция замены содержимого полей, введя данные в уже не раз упоминавшиеся графы **SCOPE**, **FOR** и **WHILE**.

Пометка записей на удаление выполняется выбором опции **Mark record for deletion**. В шаблон этой опции можно ввести только условия отбора записей для этой операции. Если эти условия не введены, то помечаются все записи файла БД. Аналогично осуществляется уничтожение этих пометок.

Для того, чтобы сгенерировать индексный файл, выбирается опция **Generate index**. После выбора этой опции высвечивается шаблон с графами для ввода следующих данных: выражение для создания индекса (**Index KEY expression**), которое может включать любое количество полей (нельзя индексировать **Memo** поля) и допустимые с ними в dBASE IV действия (кроме логических), установку на включение в индексный файл только первой записи с повторяющимся значением индекса (**Yes**) или включение всех записей (**No**) (**Index first key occurance only**), установку индекса в убывающем порядке (**Yes**) (**Index in DESCENDING order**), имя индексного файла (только для типа **.ndx**), который требуется сгенерировать (**INDEX file**), имя указателя индекса в множественном индексном файле (для типа **MDX**) (**TAG**) (если не создается индексный файл типа **.ndx**), имя **.MDX** файла, указатель которого введен в предыдущей графе (**MDX file**). Переиндексация файла БД осуществляется выбором опции **Reindex database**. Это требуется в том случае, если в файл БД были добавлены или изменены записи при закрытом индексном файле.

Сортировка записей с созданием нового файла БД назначается выбором опции **Physically sort database**. В шаблон этой опции заносятся данные: имя нового отсортированного файла (**TO file**), имена полей в порядке, определяющем сортировку записей (**Sort FIELDS**), порядок сортировки - возрастающий (**ASCENDING**) или убывающий (**DESCENDING**) (**Sort order**), различение (**USE CASE**) или не различение (**IGNORE CASE**) прописных или строчных букв. Кроме того, вводятся условия отбора записей для сортировки в упоминавшиеся стандартные графы **SCOPE**, **FOR** и **WHILE**.

Опции **Import foreign file** и **Export foreign file** позволяют назначить позиции меню прикладной программы импорт файлов из других СУБД и экспорт своих файлов БД. При назначении импорта в графы шаблона следует ввести имя файла (**From file**) и выбрать его тип (**of type**). В шаблон экспорта также нужно ввести имя файла БД (**To file**) и тип файла. Кроме того, в этот шаблон можно ввести список полей файла БД, которые перено-

сятся в экспортируемый файл (**Fields**), и условия отбора записей для экспорта (**SCOPE**, **FOR** и **WHILE**).

Кроме операций с файлами, позиции меню прикладной программы можно также назначить выполнение команд dBASE IV и DOS. Для этого служит опция **Run program**. После выбора этой опции высвечивается меню, позволяющее назначить выполнение программы (процедуры) dBASE (dBASE III и dBASE IV), написанной на языке dBASE и SQL, группы команд dBASE, команды (утилиты, программы) DOS, двоичного модуля (с расширением **.bin**), макроопределения, а также выполнение модуля **Batch**.

Для назначения выполнения программы (модуля) dBASE выбирается опция **Do dBASE program** и заполняется шаблон. В графу **Program** вводится имя dBASE программы (расширение **.dbo**, **.prg** или **.prs**), а в графу **Parameters** - параметры для этой программы. Опция **Insert dBASE code** используется для назначения позиции меню выполнения группы команд. После выбора этой опции появляется экран, на котором можно записать до 19 строк команд dBASE. Назначение выполнения команд (программ, модулей, утилит) DOS осуществляется опцией **Run DOS program**, после выбора которой высвечивается шаблон с графами для ввода имени команды (**Program**) и ее параметров (**Parameters**). Например, при назначении команды **DOS CHKDSK** можно передать параметр **/f**. Опция **Load/call binary file** служит для назначения загрузки или выполнения двоичного модуля (написанного на Ассемблере) с расширением **.bin** и имеет шаблон, аналогичный предыдущей опции.

Для назначения позиции меню выполнения созданного ранее макроопределения используется опция **Play back macro**, после выбора которой высвечивается окно для ввода имени макроопределения.

Опция **Execute BATCH process** назначает выполнение группы операций модуля **Batch**. После выбора этой позиции требуется ввести в окно имя модуля **Batch**.

В меню прикладных программ обычно предусматривается выход в DOS, УЦ dBASE IV или в вызывающую программу (процедуру). Назначение таких операций выполняется с помощью опции **Quit**. После выбора опции высвечивается меню с двумя позициями - для выхода в вызывающую программу выбирается **Return to calling programm**, а для выхода в DOS - **Quit to DOS**.

Как уже упоминалось, система меню прикладных программ может строиться в виде вложенных многоуровневых структур меню вертикального или горизонтального типа. Для реализации такой структуры используется операция открытия меню, которая назначается опцией **Open a menu**. После выбора этой опции высвечивается шаблон с двумя графами, позволяющими выбрать тип и ввести имя открываемого меню или списка. Если позиции

прикладного меню не требуется назначать какую-либо операцию, то используется опция **Text (no action)**, которая высвечивает текст позиции и не выполняет никаких действий (по умолчанию назначается всем позициям меню). Такие позиции часто используются для пояснения выполнения нижеследующих позиций вертикального меню.

Несколько опций меню **Item** позволяют, кроме основного действия, назначить еще и дополнительное действие. Например, опция **Position record pointer** дает возможность назначить установку указателя позиции записи в файле БД. После выбора этой опции высвечивается шаблон, в котором следует выбрать значения **Yes** или **No** в графе **Display POSITIONING MENU at RUN TIME?**. Если задано значение **Yes**, то при выборе позиции прикладного меню высвечивается вертикальное меню, позволяющее: найти запись по индексу, перейти к записи с заданным номером, а также к первой или последней записи, найти запись без индекса и возвратиться в предыдущее меню. Для поиска записи с требуемым значением индексированного поля выбирается позиция **SEEK Record** (этот индекс должен быть установлен главным, т.е. управляющим), после чего следует ввести поисковое значение. Чтобы перейти к записи с заданным номером (к первой или последней записи), выбирается позиция **GOTO Record** и вводится номер записи (выбираются опции **Top** или **Bottom**). Поиск по неиндексированным полям задается выбором позиции **LOCATE Record** и последующим вводом искомого значения поля (для символьных полей в кавычках). Возврат в предыдущее меню осуществляется выбором позиции **Return**.

Если в графе **Display POSITIONING MENU at RUN TIME?** выбрано **No**, то в соответствующие графы шаблона можно ввести данные, позволяющие установить указатель записи в файле БД сразу после выбора позиции меню прикладной программы посредством поиска по индексированному и неиндексированному полям или по номеру записи. Эти данные аналогичны указанным выше. В графу **SEEK first occurance of key** вводится выражение для поиска по индексированному полю, в графу **GOTO** - номер записи (или ключевые слова **Top** или **Bottom**), в графы **LOCATE** вводятся данные для поиска по неиндексированным полям: **SCOPE** - диапазон поиска, **FOR** - поисковое выражение, **WHILE** - условие конца поиска.

К числу дополнительных действий также относится назначение индексного файла или указателя индекса в множественном индексном файле (**Reassigne index order**). Для этого требуется ввести в шаблоне в графу **Set ORDER to** число, определяющее порядковый номер индекса в списке индексных файлов **.ndx** или указателя индекса в множественном индексном файле **.mdx**, а также имя **.mdx** файла в графе **OF**. Очень полезным дополнительным действием является назначение ввода и вывода данных файла БД в окно на экране. Одновременное высвечивание в

окнах данных из нескольких файлов БД обеспечивает удобство работы с ними. Для назначения вывода в окно служит опция **Define logical window**. После выбора этой опции требуется ввести в шаблон следующие данные: имя окна в графу **Window NAME**, тип рамки окна - одинарная линия (**SINGLE**), двойная линия (**DOUBLE**), сплошная полоса (**PANEL**), определяемая пользователем (**CUSTOM**) и без рамки (**NONE**) в графу **Display border as**, буквенные коды цвета стандартной и нестандартной областей экрана, рамки и фона (например, G+/N,W/R,G+ устанавливает цвет стандартных областей экрана зелеными буквами на черном фоне, нестандартные области (подсвеченные поля) - белыми буквами на красном фоне и зеленый цвет рамки) в графу **Colors**, координаты левого верхнего и правого нижнего углов окна в графы **UPPER LEFT corner at row and column** и **LOWER RIGHT corner at row and column**. Если рамка окна задается пользователем, то в графу **Border characters (for CUSTOM borders)** следует ввести символы или их коды ASCII для определения сторон рамки окна. Символы заключаются в двойные кавычки, а коды ASCII в фигурные скобки. Если устанавливаются различные символы для сторон рамки, то они отделяются запятыми.

С позицией меню прикладной программы можно связать справочный текст, который высвечивается клавишей **F1** при выборе данной позиции. Если этот текст не введен, то высвечивается текст, который назначен меню или списку в целом. При отсутствии и этого текста высвечивается сообщение **No help defined** (справочного текста нет). Справочный текст (до 19 строк) записывается после выбора опции **Write help text** на полном экране. К справочной информации относится также текст сообщения (длиной до 76 символов), высвечиваемого внизу экрана при выборе позиции. Этот текст можно ввести после выбора опции **Assigne message line prompt**. Двойные кавычки внутри текста запрещаются.

Кроме дополнительных действий можно ввести команды языка dBASE, которые будут выполняться перед или после выполнения основного действия, назначенного позицией меню прикладной программы. Эти команды записываются на полном экране, который высвечивается после выбора опции **Embed code** и соответствующего ключевого слова: **Before** (перед) или **After** (после). Для ввода большего количества команд можно предварительно создать процедурный модуль любой длины с расширением .prg и вызвать его, введя команду **DO <имя процедуры>.prg**. В частности, эта операция позволяет превратить списки в полноправные меню, когда после выбора каждой позиции списка можно выполнять различные действия. Для реализации этого можно использовать функцию **BAR()**, которая возвращает номер позиции списка. Ее следует включить в команды **DO CASE** или **IF** для возвращения номера позиции

списка в условие, определяющее ветвление процесса дальнейшего выполнения команд.

Завершает список дополнительных действий опция **Bypass Item on condition**, позволяющая задать условие, по которому операция, назначенная позиции меню, не будет выполняться после ее выбора (только для вертикального меню). В шаблон этой опции с графой **Skip this item if** требуется ввести условие отключения выполнения операции, назначенной позиции меню. Например, можно для позиции, которая обрабатывает требуемый файл БД, установить условие **.NOT. <имя файла>**, означающее, что данная позиция пропускается, если выбирается не тот файл.

Если при выборе позиции меню нужно обрабатывать файл БД, отличный от назначенного в главной процедуре или назначенного конкретно для этого меню, то для изменения сделанного ранее назначения используется опция **Override assigned database or view**. Изменение назначения можно выполнить двумя способами: заменить имена файла БД и связанных с ним индексных файлов на новые или же задать возможность ввода имен файлов при выборе позиции меню прикладной программы в ходе ее выполнения. Для этого нужно в графе **For this item you may use values** выбрать одно из трех ключевых слов: **IN EFFECT AT RUN TIME, ABOVE, ENTERED BELOW**. Первое ключевое слово задает возможность назначения файла БД в ходе выполнения прикладной программы, второе ключевое слово оставляет назначение файла БД, сделанное ранее, а третье позволяет сделать новое назначение. В последнем случае требуется заполнить следующие графы шаблона: имя файла БД (**Database/view**), имена индексных файлов (**Set INDEX to**), порядковый номер главного (управляющего) индекса или его имя (**ORDER**).

Опция **Show item information** меню **Item** высвечивает сведения об основном действии, назначенном позиции меню, в следующих графах: имя меню (**Object**), имя позиции меню (**Item**), имя файла БД (**Current database/view**) и индексного файла (**using index**), назначенного данной позиции. Если этой позиции (или меню) специально не назначался файл БД (индексный файл), то высвечивается назначение, принятое в главной процедуре. В графе **This item will** высвечивается основное действие, которое назначено данной позиции.

Позиция **Generate** основного меню генератора прикладных программ генерирует программу и создает для нее документацию. В процессе генерации программы создаются два файла с именами: **<имя главной процедуры>.prg** и **<имя главного меню>.prg**. Их следует хранить в одном директории для последующего запуска. Файлу документации присваивается имя главной процедуры и расширение **.doc**. После выбора позиции **Generate** высвечивается вертикальное меню с опциями: **Begin generating** - генерация программы или создание документации в

зависимости от выбора файла генерации (шаблона) в следующем меню, **Select template** - назначение шаблона для создания программы или документации. По умолчанию принимается шаблон **MENU.GEN** для генерации программы, но можно выбрать другие шаблоны - **DOCUMENT.GEN** для создания документации или **QUICKAPP.GEN** для генерации стандартной прикладной программы (при этом текущим компонентом должна быть главная процедура, т.е. в основном меню должна высвечиваться позиция **Application**).

Опция **Display during generation** позволяет высвечивать программу или документацию в ходе их генерации. После генерации документации или программы их можно отпечатать в режиме ввода команд с клавиатуры командой **Type <имя программы или документации> on printer** или из меню **Print** текстового редактора.

Позиция **Exit** основного меню генератора прикладных программ служит для выхода в УЦ с сохранением выполненных изменений или без сохранения и имеет прикрепленное вертикальное меню с двумя опциями. Первая опция - **Save all changes and exit** - сохраняет все компоненты прикладной программы, находящиеся в памяти, и прекращает сеанс работы с генератором. Обычно промежуточное сохранение изменений с целью последующей работы над прикладной программой выполняется с помощью позиции **Save current** в меню разработки компонентов прикладной программы **Menu**, **Batch** или **List**. Можно сохранить разработанную прикладную программу под другим именем для последующей модификации или адаптации применительно к другим задачам. Вторая опция - **Abandon all changes and exit** - обеспечивает выход без сохранения последних изменений. На диске остаются ранее сохраненные версии прикладной программы. Для подтверждения выхода без сохранения требуется выбрать **Yes**, после чего выполняется возврат в УЦ или в командный режим в зависимости от того, откуда был осуществлен вход в генератор прикладных программ. Если выбирается **No**, то осуществляется возврат в основное меню генератора без изменения состояния разрабатываемых компонентов.

После того как прикладная программа создана и отлажена, ее можно запустить из УЦ, выбрав ее имя в окне генератора. После этого высвечивается меню с позициями на запуск программы (**Run programm**) или ее модификации (**Modify programm**).

Для закрепления прочитанного материала попробуйте с помощью генератора прикладных программ создать простую программу, позволяющую пользователю работать с файлами БД нашего примера и выполнять действия по вводу и обновлению записей, отбору записей и выводу на печать отчета. В программу будут включены ранее рассмотренные файлы экранных форм **MF1**, **MF2**, **MF3**, **MF4**, запросы **MQ15**, **MQ16** и отчет **MR1**.

Прикладная программа состоит из главной процедуры и вызываемых модулей. Сначала нужно создать главную процедуру. Для этого выберите позицию **<create>** окна **Applications**, а в высветившемся окне - позицию **Applications Generator** и нажмите **Enter**. На экране появится шаблон **Application definition** для описания главной процедуры, в который нужно ввести приведенные ниже данные:

Application name (имя главной процедуры) - **MAVTO**;
Description (описание) - Программа ведения БД;
Main menu type (тип главного меню) - **POP-UP**;
Main menu name (имя главного меню) - **MP1**;
Database/view (файл БД/выборка) - **MBD1.**

Все данные вводятся с клавиатуры, за исключением типа главного меню, которое выбирается клавишей **Пробел**. Для сохранения данных о главной процедуре нажмите клавиши **Ctrl-End**. После сохранения данных высвечивается стандартная заставка с текстом, начинающимся с фразы **This is an APPLICATION OBJECT...**, его можно стереть и ввести свой, подобный показанному ниже.

```
Прикладная программа  AVTO

Дата создания: 25.09.1989 г.

Разработчики: Б.Романов, А.Кушниренко
```

Для того чтобы окно такого типа появлялось при запуске Вашей программы **MAVTO**, нажмите клавишу **F10**, в меню **Application** выберите позицию **Display sign-on banner** и в опции **Display frame as "sign-on banner" at run time** (высветить на экране заставку при запуске программы) введите **YES**.

После разработки прикладной программы можно сгенерировать на нее документацию, в которой содержатся данные об авторах программы, права на копирование и версия пакета dBASE IV. Эту информацию можно также ввести в заставку разрабатываемой программы (заметим, что данные вводятся не в текущую, а только в последующую разрабатываемую программу). Сначала выберите опцию **Edit program header comments** меню **Application**, после чего высветится окно с позициями **Application author, Copyright notice** и **dBase**, в которое введите данные: автор программы (**Application author**); авторские права (**Copyright notice**), например **"Разрешено копирование"**; версия dBASE (**dBASE version**), например **"dBASE IV версия 1.0"**, и нажмите клавиши **Ctrl-End**.

Для сохранения главной процедуры выберите позицию **Save current application definition** меню **Application**, которое, напомним,

как и все меню активизируется клавишей **F10**. Теперь создайте вертикальное (**Pop-up**) меню прикладной программы, в котором будут записаны названия выполняемых операций. Для этого перейдите в меню **Design** и выберите позицию **Pop-up**. Справа на экране высветится окно имен вертикальных меню. Выберите позицию **<create>** и нажмите клавишу **Enter**. Появится шаблон для ввода данных об этом меню: имени, описания и сообщения, которое будет высвечиваться внизу экрана при переходе на это меню. Введите в графу **Name** (имя) - **MP1**, а в графу **Description** (описание) - **"Вертикальное меню"**. В графу **Message line prompt** (сообщение) можно ничего не вводить. Для сохранения описания вертикального меню нажмите клавиши **Ctrl-End**.

В центре экрана появится вертикального меню с пустыми позициями. Нажав клавиши **Shift-F7**, можно навигационными клавишами изменить размеры рамки меню так, чтобы ввести в него текст позиций, показанный ниже (новые размеры нужно зафиксировать клавишей **Enter**):

```
 ------------------------------
|         БАЗЫ  ДАННЫХ         |
|------------------------------|
|   Информация о водителях     |
|   Автомобили                 |
|   Перевозки грузов           |
|   Технические характеристики |
|==============================|
|   ОТБОР данных по перевозкам |
|------------------------------|
|   ВЕДОМОСТЬ расхода горючего |
|==============================|
|       Выход из задачи        |
 ------------------------------
```

Позиции меню, составляющие смысловые группы, для улучшения восприятия можно отделить двойными или одинарными линиями, которые рисуются так же, как и при разработке марок: при нажатии клавиш **Alt-205** - двойная линия и **Alt-196** - одинарная линия. После того как введен текст позиций меню, им нужно назначить действия, которые будут выполняться при выборе этих позиций. Например, требуется, чтобы при выборе позиции "**Информация о водителях**" был открыт файл БД **MBD1**, подключен файл экранной формы **MF1** и высвечивалось содержимое полей БД в этой форме. Для этого установите курсор в позицию "**Информация о водителях**" и выберите позицию **Override assigned databases or view** меню **Item**, в высветившемся окне введите имя файла БД, который будет использоваться при выборе данной позиции меню. Переместите курсор в позицию окна **For this item you may use values** и нажатием клавиши **Пробел** выберите опцию **ENTERED BELOW**, затем в нижней позиции **Database/view** введите имя файла - **MBD1**, а в позиции

ORDER имя главного индекса - **FAM** и нажмите **Ctrl-End** для сохранения введенных данных.

После установки файла БД нужно назначить выполняемое с этим файлом действие. Допустим, что для данной позиции нужно назначить просмотр информации в ранее разработанной экранной форме **MF1**. Для этого выберите позицию **Change action**, затем в высветившемся вертикальном меню - позицию **Edit form (add, delete, edit)**. Появится шаблон для ввода параметров команды **Edit/Append**. В графу **FORMAT file** введите **MF1** (имя файла экранной формы), в графе **Mode** с помощью клавиши **Пробел** выберите опцию **Edit** (разрешено редактирование). Другие значения в этом шаблоне менять не надо. Для сохранения всех установленных в шаблоне значений опций нажмите клавиши **Ctrl-End**.

Кроме действий, каждой позиции меню можно назначить высвечивание в нижней строке экрана сообщения, которое будет появляться при перемещении на нее курсора при эксплуатации сгенерированной прикладной программой. Для этого выберите позицию **Assign message line prompt** меню **Item**, после чего высветится окно, в которое с клавиатуры введите текст, например "**Информация из MBD1**", и нажмите клавиши **Ctrl-End**.

Для назначения действий следующей позиции нужно переместить на нее курсор. Перемещение по вертикальному меню осуществляется навигационными клавишами **Вверх** и **Вниз**. Для этого нужно клавишей **Esc** убрать меню **Item**, установить курсор на нужную позицию меню и снова активизировать меню клавишей **F10**.

Есть другой, более короткий способ перемещения курсора, без удаления меню **Item** - с помощью клавиши **PgDn**. При этом название текущей позиции создаваемого меню можно увидеть в нижней строке экрана. Для позиции "**Автомобили**" выполните те же операции по открытию файла **MBD2**, установке главного индекса **N_Z**, подключению экранной формы **F2** и назначению высвечивания сообщения в нижней строке экрана "**Информация из MBD2**".

Для позиции меню "**Перевозки грузов**" будут использованы файл **MBD3**, главный индекс **N_Z**, файл экранной формы **MF3** и информационное сообщение "**Информация из MBD3**", а для позиции "**Технические характеристики**" - файл БД **MBD4**, главный индекс **MAR**, экранная форма **MF4** и информационное сообщение "**Информация из MBD4**".

При назначении действий позиции меню "**ОТБОР данных по перевозкам**" нужно установить выполнение запроса на отбор данных по перевозкам, который ранее был сформирован под именем **MQ15**, и высвечивание данных в форме **Browse**. Для этого выберите позицию **Override assigned databases or view** меню **Item**, в опции **For this item you may use values** установите **ENTERED BELOW**, а в нижней позиции **Databases/view** введите

имя файла запроса на выборку **MQ15.qbe** и нажмите **Ctrl-End**. Заметим, что имя выборки надо обязательно указывать с расширением **.qbe**. В противном случае имя выборки будет воспринято как имя файла БД, в результате чего высветится сообщение, что такой файл не найден. Затем, как и для предыдущей позиции, нужно назначить форму вывода данных. Пусть это будет стандартная форма **Browse**. Для этого выберите позицию **Change action** меню **Item**, а из высветившегося меню - позицию **Browse (add, delete, edit)**. Установленные по умолчанию параметры команды **Browse** изменять не надо, поэтому сразу нажмите **Ctrl-End**. Можно также назначить этой позиции информационное сообщение "**Отбор данных по перевозкам на расстояние свыше 300 км**", отражающее содержанию запроса **MQ15**.

Позиции "**ВЕДОМОСТЬ расхода горючего**" нужно назначить вывод отчета **MR1**. Для этого в меню **Item** выберите позицию **Change action**, в высветившемся вертикальном меню - позицию **Display or print**, а затем позицию **Report** в меню следующего уровня. Появится окно-шаблон с графами для установки ряда параметров отчета. В позицию **FORM** введите имя созданного файла отчета **MR1**, в позиции **Send output to** выберите клавишей **Пробел** параметры **SCREEN** (экран), **DISK FILE** (файл на диске) или **PRINTER** (принтер) в зависимости от того, куда Вы собираетесь выводить сгенерированный документ, в позиции **Report format** выберите значение **Full detail** (для вывода всех записей) и в позиции **Before printing** - значение **DO NOT EJECT** (отмена перевода страницы перед печатью). Кроме этого, нужно назначить файл или выборку, данные из которых будут выводиться в отчет. Для этого, как и при назначении в предыдущей позиции, выберите позицию **Override assigned databases or view** меню **Item**, в опции **For this item you may use values** установите клавишей **Пробел** значение **ENTERED BELOW**, а в поле **Databases/view** введите имя файла запроса на выборку **MQ16.qbe** и нажмите **Ctrl-End**. Как и для предыдущих позиций можно назначить информационное сообщение, например "**Выводится отчет MR1**".

Последовательность операций по назначению действия последней позиции "**Выход из задачи**" аналогична предыдущей. Сначала установите курсор на эту позицию разрабатываемого меню, затем выберите позицию **Change action** меню **Item** генератора прикладных программ, а в высветившемся меню второго уровня - позицию **Quit** (выход из программы). Наконец, в меню третьего уровня определите, куда возвращаться. Для возврата в вызвавшую программу выберите позицию **Return to calling programm** - в нашем случае это УЦ. После выбора этой позиции высветится окно, в котором нужно подтвердить назначение клавишами **Ctrl-End** или отказаться от него клавишей **Esc**. Для этой позиции также не будет лишним назначить информаци-

онное сообщение, поясняющее, куда передается управление после завершения работы с прикладной программой "**Выход в Управляющий Центр**".

После того как вертикальное меню сформировано, надо сохранить введенные данные по его определению, выбрав позицию **Save current menu** меню **Menu**. После этого можно генерировать прикладную программу. Сначала клавишами **F3/F4** установите курсор на заставку прикладной программы. Затем выберите позицию **Begin generating**. Начнется генерация текста прикладной программы, который будет записан в файл с именем **MAVTO** и расширением **.prg** (**MAVTO.prg**). Для выхода из генератора прикладных программ выберите позицию **Save all changes and exit** меню **Exit**.

Запуск созданной программы осуществляется выбором имени программы в окне **Applications** панели УЦ. После этого высвечивается горизонтальное меню с позициями **Run application** (Запуск программы) и **Modify application** (Модификация программы), из которых нужно выбрать первую. Затем высвечивается предупреждение **Are you sure you want to run this application?** (Вы действительно хотите запустить эту программу?) с позициями **Yes** и **No**. Выбор **Yes** осуществляет запуск программы, а **No** возвращает управление на панель УЦ.

2.10. РЕДАКТОР ТЕКСТОВ

Пакет dBASE IV имеет встроенный редактор текстов, который в основном применяется для разработки прикладных программ (процедур, модулей), а также для ввода текстовых материалов. Редактор текста также используется при создании различных файлов и вызывается автоматически из различных точек УЦ, в том числе при вводе текстовых данных в поля **Memo** файлов БД, при разработке экранных форм и отчетов. При создании программ или вводе текста редактор обычно вызывается из УЦ в окне генератора прикладных программ. Для этого сначала нужно выбрать позицию **<create>** в окне **Applications** панели УЦ. После этого высвечивается меню с двумя позициями: **dBASE program** и **Application Generator**. Первая вызывает редактор текста, а вторая открывает экран разработки прикладной программы с помощью генератора.

Сохранение введенного текстового материала осуществляется с помощью меню редактора, которое описано ниже. Здесь отметим, что по умолчанию всем текстовым файлам присваивается расширение **.prg**. Однако в dBASE IV для текстовых файлов разрешается использовать любые расширения, кроме, естественно, принятых для других типов файлов. Например, нельзя давать расширения **.dbf**, **.mdx**, **.frm** и др., поскольку текстовый файл с такими расширениями будет высвечиваться на панели УЦ

в соответствующем окне, а при его запуске возникнет ошибочная ситуация. Во многих пакетах обработки информации в качестве расширения для текстовых файлов принято **.txt**. Это расширение стало стандартом для текстовых файлов. Поэтому в СУБД dBASE IV для отличия чисто текстовых файлов от программных целесообразно использовать именно это расширение.

В качестве примера в учебном каталоге **PRIMER** введены два учебных текстовых файла **TEXT1.txt** и **TEXT2.txt**, которые читатель может выбрать в окне **Applications** панели УЦ. После выбора имени одного из этих файлов высвечивается горизонтальное меню с позициями **Run application** и **Modify application**. Первая позиция запускает на выполнение прикладную программу, которая записана в вызванном текстовом файле, а вторая служит для ввода текста или просмотра содержимого файла.

Редактор текста имеет основное меню с позициями: **Layout**, **Words**, **Goto**, **Print**, **Exit**. К позиции **Layout** прикрепляется вертикальное меню, позволяющее вызвать вместо текущего другой текстовый файл с помощью позиции **Modify a different program**. После выбора этой позиции высвечивается окно с предложением (**Enter program to modify**) ввести имя другого текстового файла. Это имя файла можно набрать с клавиатуры или нажать клавиши **Shift-F1** и выбрать имя файла из высвеченного списка файлов.

Заметим, что если текстовых файлов в текущем каталоге нет (в нашем случае это файлы с расширением **.prg** или **.txt**), то после вызова списка клавишами **Shift-F1** в нижней части экрана высвечивается соответствующее сообщение. Кроме того, в каталог можно ввести словесное описание создаваемой программы (**Edit description of program**).

Позиция **Save this program** сохраняет файл на диске. Перед сохранением высвечивается окно **Save as** для ввода нового или подтверждения старого имени. Это дает возможность сохранить копию файла под другим именем, чтобы потом использовать для модификации. Как и при сохранении других типов файлов, в окне перед именем файла нужно поместить имя диска и маршрут по директориям.

Позиция **Print** имеет прикрепленное меню, в основном аналогичное меню разработки отчетов. Отличие состоит в том, что в данном меню имеется дополнительная опция **Line numbers**, позволяющая печатать номер строки в первых позициях текста.

К позиции **Exit** прикреплено меню с позициями (первые две стандартные): сохранение файла на диске (**Save changes and edit**) и выход без сохранения (**Abandon changes and edit**) и несколькими нестандартными, специфичными для редактора текста. К последним относятся: сохранение выполненных изменений и запуск программного файла на выполнение (**Run program**), сохранение программного файла и запуск отладчика с

последующим высвечиванием экрана для задания его параметров, включая точки прерывания и параметры, передаваемые отладчику (**Debug program**).

Позиции **Words** и **Goto** в основном совпадают с аналогичными позициями в меню разработки экранных форм и генератора отчетов.

2.11. РАЗГРАНИЧЕНИЕ ДОСТУПА К ИНФОРМАЦИИ И ЗАЩИТА ДАННЫХ

Для создания средств защиты от несанкционированного входа в dBASE IV и разграничения доступа к информации используется позиция **Protect data** меню **Tools**. Средства защиты включает три типа:

защиту от несанкционированного входа в пакет dBASE IV с использованием паролей пользователей;

защиту доступа к файлам и полям БД;

шифрование данных в файлах БД.

Комплекс этих средств образует систему защиты информации. Пользователи в системе защиты информации объединяются в группы. Каждый пользователь может входить в состав нескольких групп и иметь в них различные уровни доступа к информации. Обычно группа связывается с множеством файлов, которые могут обрабатываться этой группой. Один и тот же файл может входить только в одно множество, связанное с определенной группой.

Вход в пакет осуществляется после ввода имени пользователя, имени группы, в состав которой входит пользователь, и пароля (при вводе на экране не отображается). Если пользователь входит в состав нескольких групп и ему требуется получить доступ к файлам БД другой группы, для этого необходимо сначала выйти из системы и затем войти снова в составе новой группы. Доступ к файлам БД и полям осуществляется в соответствии с присвоенным пользователю общим уровнем доступа, который определяет привилегии по использованию данных. Уровни доступа от высшего к низшему нумеруются от 1 до 8. Для каждого файла, связанного с определенной группой пользователей, назначается уровень доступа при установлении схемы привилегий использования конкретных файлов и полей.

После входа в dBASE IV определяется фактический уровень доступа пользователя к конкретному файлу или полю из группы посредством сопоставления его общего уровня доступа с уровнем доступа к этому файлу из схемы привилегий. Если уровень доступа к файлу или его полю из схемы привилегий выше общего уровня доступа пользователя, то данная привилегия ему недоступна. Схема привилегий для использования файлов БД в группе устанавливается посредством назначения уровней

доступа к различным операциям с файлами БД. Различают четыре типа операций:

чтение - просмотр записей в файлах БД;

обновление - изменение содержимого полей файлов БД;

добавление - добавление новых записей в файлы БД;

удаление - удаление записей из файлов БД.

Если привилегии по использованию файлов БД не установлены, то по умолчанию разрешаются все операции.

При обработке полей различаются три группы привилегий:

разрешены все операции - чтение и запись данных в поля (устанавливается по умолчанию);

разрешено только чтение - чтение (просмотр) содержимого поля без его изменения;

запрещены все операции - нельзя просматривать или изменять содержимое поля (доступ к полю блокируется).

При использовании средств защиты информации все файлы БД, включая связанные индексные файлы и поля **Memo** (файлы), автоматически шифруются и хранятся на диске в зашифрованном виде. При выводе выполняется их автоматическое дешифрование.

Средства защиты информации в dBASE IV поддерживают два системных зашифрованных файла: **Dbsystem.db** и **Dbsystem.sql** (в зависимости от типа используемого командного языка - dBASE IV или SQL), которые содержат данные по системе защиты, включая имя пользователя, учетное имя пользователя, пароль, групповое имя и общий уровень доступа. Доступ к этим файлам осуществляется по специальному паролю, известному администратору системы. Этот пароль назначается один раз при установке системы dBASE IV.

Данные по защите информации и доступу пользователей устанавливаются и изменяются администратором. Средства защиты могут использоваться в однопользовательском режиме и обязательно в многопользовательском режиме работы dBASE IV, в том числе и при работе в среде ЛВС. При первоначальном выборе позиции **Protect** высвечивается окно для ввода пароля администратора системы. После ввода пароля его требуется подтвердить, затем раскроется меню для ввода данных по доступу пользователя. Это меню имеет позиции **Users**, **Files**, **Reports**, **Exit**.

Позиция **Users** позволяет ввести все общие данные на пользователя по системе защиты информации в соответствующие графы таблицы: **Login name** - имя пользователя, **Password** - пароль пользователя, **Group name** - имя группы, **Full name** - учетное имя пользователя, **Acces level** - общий уровень доступа. Опция **Store user profile** позволяет сохранить данные по доступу пользователя в соответствующем файле. Если предполагается работа с языком SQL, то необходимо ввести имя пользователя-администратора **SQLDBA**, имеющего доступ ко всем таблицам.

При вводе уже имеющейся комбинации имени пользователя, пароля и имени группы высвечивается предупреждение и предложение сделать изменения. Нельзя изменять или удалять имя группы. В этом случае файлы данной группы становятся недоступными всем пользователям. Для удаления пользователя требуется набрать на клавиатуре его комбинацию доступа в систему и затем после выбора опции редактирования выбрать позицию **Delete user from group**. После подтверждения пользователь удаляется из группы.

Позиция **Files** меню **Protect** позволяет создать или обновить схему привилегий по доступу к файлам и полям в группах. Сначала нужно выбрать файл БД с помощью опции **New file** из высвечиваемого списка в текущем или другом директории. Затем в опции **Group name** задается имя группы. В опции **File access privileges** назначаются уровни доступа (от 1 до 8) операциям по обработке файла: чтения (**Read privilege**), обновления данных (**Update privilege**), добавлению записей (**Extend privilege**) и удалению записей (**Delete privilege**). Назначение привилегий по доступу к полям выполняется опциями позиции **Field access privileges**. Опцией **Access level** устанавливаются привилегии обработки поля: разрешены все операции (**FULL**), разрешено только чтение (**R/O**), запрещены все операции (**NONE**). Опцией **Establish field privileges** назначаются уровни доступа (от 1 до 8) для выбранной привилегии.

Для сохранения установленных привилегий файла, входящего в состав группы, выбирается опция **Store file privileges**, а для отмены назначений текущего сеанса - **Cancel current entry**. Первоначальное назначение уровней доступа и схем привилегий выполняется сверху вниз по опциям вертикального меню. Изменение схемы привилегий осуществляется в том же порядке. При сохранении изменений высвечивается сообщение, напоминающее о существовании старой схемы, и предложение перезаписать схему. После подтверждения (**Y**) изменения сохраняются. После этого файлы БД и поля становятся защищенными и создаются копии зашифрованных файлов с расширением **.crp**. При этом исходные файлы с расширением **.dbf** сохраняются. Установленная схема доступа к файлу записывается в структуру файла БД. Если уровни доступа и схемы привилегий для файлов и полей не установлены, то в системе защиты будут задействованы только данные по входу пользователей в систему.

Для гарантированной защиты информации после создания зашифрованных копий файлов следует удалить исходные файлы БД. Целесообразно их предварительно скопировать на гибкие диски и хранить копии в защищенном месте. Чтобы система могла работать с зашифрованными файлами, их целесообразно сначала скопировать на место исходных файлов, а затем изменить расширение **.crp** на **.dbf**. После этого надо удалить исходные зашифрованные файлы (с расширением **.crp**). Кроме

того, следует заменить расширение **.cpt** файлов **Memo** полей на расширение **.dbt**. Индексные файлы шифруются в том случае, если они созданы вместе с зашифрованными файлами или при выполнении команды **REINDEX** с этими файлами. При выполнении команды **PROTECT** по умолчанию устанавливается команда **SET ENCRYPTION ON** и копии файлов также шифруются. При установке **SET ENCRYPTION OFF** копии не шифруются.

Позиция **Reports** меню **Protect** позволяет (опция **User information**) вывести на экран информацию по доступу пользователей, которую можно также распечатать, если ответить **(Y)** на предложение **Send report to the printer? (Y/N)**. Опция **File information** выводит на экран информацию об уровнях и схемах привилегий доступа к файлам и их полям. Выбор этой опции высвечивает список защищенных и зашифрованных файлов БД. После выбора требуемого файла БД можно вывести на принтер информацию о доступе к нему.

Позиция **Exit** включает опции, позволяющие сохранить все изменения, сделанные в ходе текущего сеанса (**Save**), аннулировать все изменения, выполненные в ходе текущего сеанса (**Abandon**) и прекратить текущий сеанс установки схемы доступа пользователя с сохранением всех сделанных назначений (**Exit**).

Глава 3
КОМАНДЫ И ФУНКЦИИ ЯЗЫКА dBASE IV

3.1. ОБЩИЕ СВЕДЕНИЯ О КОМАНДАХ

Описанный в гл. 2 УЦ позволяет выполнять многие операции без знания командного языка dBASE IV с помощью режима, управляемого меню. Однако для более полного и грамотного использования возможностей УЦ начинающим пользователям целесообразно изучать командный язык dBASE IV. Это, естественно, относится и к пользователям, которые не удовлетворяются тем, что дает УЦ, и хотят более глубоко освоить пакет dBASE IV. Кроме того, командный язык позволяет глубже понять принципы работы пакета и овладеть программированием задач, являющимся высшим уровнем освоения СУБД.

В данной главе описываются команды и функции dBASE IV, объединенные в функциональные группы примерно в той последовательности выполняемых ими действий, которая соответствует отмеченным в гл. 2 этапам технологии разработки информационных задач. Команды будут сопровождаться примерами, которые в основном ориентированы на их выполнение в

командном режиме. После овладения командным языком можно перейти к главе 5, где на конкретных примерах показывается, как создавать программы на языке dBASE IV.

Как уже отмечалось, команды языка dBASE IV можно выполнять по-одной в командном режиме или в пакетном режиме в виде программ. В состав элементов командного языка входят **команды**, **SET-команды**, **функции**, **переменные** и **массивы**.

Команда записывается в виде строки слов, в которую входит **глагол** - основное слово, определяющее содержание операции, выполняемой командой, **список выражений** и **ключевые слова** с аргументами. Структура командной строки называется синтаксисом команды. Многие команды обработки файлов строятся по следующей форме:

<глагол> [<список выражений>] [<границы>]
[FOR<условие>] [WHILE<условие>]
[TO FILE <имя файла> /TO PRINTER
/TO ARRAY <имя массива>
/TO <переменная>]
[ALL [LIKE/EXCEPT <шаблон>]] [IN <алиас>]

Квадратные скобки означают использование заключенных в них элементов синтаксиса по выбору. В угловых скобках, следующих после ключевого слова, записываются его аргументы. Косая черта, наклоненная вправо (/), обозначает, что может быть записано только одно из разделенных этим знаком ключевых слов. Угловые, квадратные скобки и косая черта в записи команды для ее выполнения не указываются и служат только для пояснения синтаксиса.

Выражение представляет собой комбинацию имен полей файлов БД, переменных, элементов массивов, констант, функций, системных переменных и операторов. Данные в полях, переменных, массивах, функциях и константы могут быть четырех типов и обозначаются в синтаксисе команд следующими сокращениями: **<выр.C>** - тип **C** (символьный), **<выр.N>** - тип **N** (числа в двоично-десятичном коде), **<выр.F>** - тип **F** (числа в двоичном коде с плавающей запятой), **<выр.D>** - тип **D** (даты), **<условие>** - тип **L** (логический). Данные символьного типа должны заключаться в разделители, например двойные или одинарные кавычки или квадратные скобки. Различие между двумя типами чисел определяется способом внутреннего представления и не влияет на форму их ввода и вывода. Существуют две основные формы ввода-вывода чисел: обычная, в виде десятичных цифр, где дробная часть отделяется точкой и в виде мантиссы с порядком числа (научная запись чисел). Числа в двоично-десятичном коде не округляются и поэтому их целесообразно использовать в финансовых расчетах, где требуется точный баланс операций. Форма чисел с плавающей запятой больше подходит для научных расчетов, так как позволяет оперировать

с очень большими и очень малыми числами, особенно при многократных вычислениях. По умолчанию все вводимые числа представляются в двоично-десятичном коде. Для перевода в форму с плавающей запятой можно использовать функцию **FLOAT()**. Если поле файла БД определено как тип **F**, то числа можно вводить в этой форме. Тип числа, возвращаемого функциями, зависит от вида функции. Так, функции **EXP()**, **LOG()**, **SQRT()** и все тригонометрические функции возвращают числа в форме типа **F**. Все другие функции возвращают тип **N** или тот тип, который использовался при вводе данных. Следует отметить, что результат совместных операций с числами типа **F** и **N** всегда представляется в форме **F**. Очень большие и очень малые числа типа **N** всегда выводятся в форме мантиссы с порядком, например, **3.655 E+09**. Данные типа даты занимают **8** байт. По умолчанию даты вводятся в американском формате **мм/дд/гг** (**мм** - месяц, **дд** - день, **гг** - год). Другой формат, например, европейский **дд/мм/гг**, можно установить командой **SET DATE** или в качестве значения параметра **DATE** в **Config.db** файле. При вводе данных типа даты для идентификации они должны заключаться в фигурные скобки, например: **{23/10/89}**. Можно вводить дату в виде символьной строки, преобразуя ее в форму типа даты специальной функцией: **CTOD("23/10/89")**. С датами можно выполнять некоторые арифметическме операции. Например, в результате вычитания дат получается разность - число дней между датами. Это число можно прибавлять или вычитать из даты, в результате чего получается новая дата.

Логические данные имеют значения **"Истина"** и **"Ложь"**, которые записываются в виде (**.T.**) и (**.F.**). При вводе допускаются альтернативные обозначения **.t.**, **.Y.**, **.y.** и **.F.**, **.f.**, **.N.**, **.n.** Логическое выражение называется также условием.

Все данные в выражении должны быть одного типа. Если в выражении требуется использовать данные разных типов, их надо преобразовать в один тип специальными функциями. Выражения в списке могут быть различного типа.

Операторы в выражениях обозначают операции с данными и обозначаются специальными символами. В dBASE IV используются четыре типа операторов: математические, операторы отношений, логические и строковые. Исходными данными и результатом действия математических операторов являются числа. В качестве символов математических операторов используются:

+	(сложение или положительный знак числа);
-	(вычитание или отрицательный знак числа);
*	(умножение);
/	(деление);
** или ^	(возведение в степень);
()	(для образования групп операторов).

Операторы отношений используются с данными любого типа. Результатом действия этих операторов являются логические значения **.T.** и **.F.** В качестве операторов отношений служат символы:

< (меньше чем);
> (больше чем);
= (равно);
<> или # (не равно);
<= (меньше или равно);
>= (больше или равно);
$ (сравнение подстрок).

Последний оператор сравнивает левую и правую строки, определяя, содержится ли левая строка в правой. Например, для строк **A** и **B** выражение **A $ B** возвращает **.T.**, если **A** = **B** или **A** содержится в **B**. С помощью операторов отношений формируются элементарные логические условия, включающие два выражения одного типа данных и оператор между ними.

Для сравнения выражений и элементарных логических условий используются логические операторы:

.AND. (логическое И);
.OR. (логическое ИЛИ);
.NOT. (логическое НЕ);
() (для образования групп логических условий).

Строковые операторы (+) и (-) позволяют сцепить две строки данных и образовать одну. При использовании первого оператора конечные пробелы обеих строк помещаются в начало, а при использовании второго - в конец результирующей строки.

В сложных выражениях, включающих несколько операторов, они выполняются в порядке, определяемом приоритетами (правилами предшествования). Операторы отношений и строковые операторы имеют равные приоритеты (один уровень предшествования) и выполняются в порядке слева направо. Математические операторы имеют разные приоритеты и выполняются в следующем порядке: знаки чисел, возведение в степень, умножение и деление, сложение и вычитание. Логические операторы выполняются в порядке: **.NOT.**, **.AND.**, **.OR.** Если в одно выражение входят различные типы операторов, то они выполняются в порядке: математические или строковые, операторы отношений, логические операторы. Операторы с равным приоритетом выполняются в порядке их записи слева направо. Для изменения порядка выполнения операторов образуются группы, помещаемые в скобки. Группы могут быть вложенными. В этом случае выполнение начинается с самых внутренних групп.

Если в выражение вводится поле из файла БД не из текущей рабочей области, то перед ним нужно указать алиас (псевдоним) этого файла БД со стрелкой вправо (->), например, **Kadry->Fam** (понятие алиаса дается ниже).

Границы в приведенном выше общем синтаксисе команды указывают те записи файла БД, которые будут доступны при его обработке. В качестве ключевых слов границ могут быть: **RECORD <n>** - доступна только одна запись с заданным номером; **NEXT <n>** - доступны все записи, начиная с номера **n**; **ALL** - доступны все записи файла БД; **REST** - доступны записи, начиная с текущей до конца файла.

Опции **FOR** и **WHILE** определяют условия отбора записей для обработки. Эти опции имеют приоритет перед границами при определении доступа к записям. Опция **FOR** определяет доступ к обработке начиная с первой записи и до конца файла БД, а **WHILE** - начиная с текущей записи и до тех пор, пока условие имеет логическое значение "**Истина**" (**True** или **.T.**).

Опция **TO** управляет выводом информации, обработанной командой. В ряде команд выходной поток можно направить в файл на диске, на принтер, в массив или переменную. Переменная представляет собой хранилище данных любого типа в оперативной памяти ПК. Для обращения к переменной ее нужно определить - назначить имя и тип. Тип определяется присвоением переменной значения соответствующего типа. Переменные могут входить в синтаксис ряда команд, например, таких как **ACCEPT**, **AVERAGE**, **CALCULATE**, **COUNT**, **INPUT**, **PARAMETERS**, **PRIVATE**, **PUBLIC**, **STORE**, **SUM** и **WAIT**. Массив представляет собой набор переменных, которые в dBASE IV могут быть представлены в виде строки (одномерный массив) или таблицы (двумерный массив) значений. Каждое значение набора переменных называется элементом массива и может входить в состав выражения. Опция **ALL LIKE** позволяет отобрать среди имен файлов, полей файлов БД и переменных те, которые соответствуют заданному шаблону. Шаблон строится по следующему принципу. В него может входить общая часть имен нескольких файлов, например, их расширение и знаки замещения специфической части **?** и *****. Первый из них замещает один, а второй любое количество символов имени. В шаблоне можно использовать только знаки замещения. Например, шаблон ***.*** определяет все имена и типы файлов. Опция **ALL EXCEPT** действует противоположно опции **LIKE** - отбирает имена за исключением указанных в шаблоне. Опция **IN** позволяет обращаться к файлу БД в другой рабочей области, не выбирая ее в качестве текущей. Аргументом этой опции может быть алиас рабочей области, ее номер или буква. Напомним, что перед тем, как манипулировать с файлом, его надо открыть. В dBASE IV разрешается одновременно открывать файлы в **10** рабочих областях. Расположение открытого файла БД отслеживается по алиасу, который представляет собой псевдоним файла БД или его имя (используется по умолчанию, если псевдоним не задан), номер или букву рабочей области. Псевдоним задается опцией **ALIAS** команды **USE** и используется обычно вместо имени

файла для ссылки на рабочую область, в которой этот файл обрабатывается. Рабочие области имеют номера от **1** до **10**, которым соответствуют буквы от **А** до **J**. Не следует в качестве имени файла использовать одну букву, так как буквы от **А** до **J** зарезервированы под имена алиасов, а некоторые из остальных используются пакетом dBASE IV в других целях. Ссылка на алиас, как и на имя файла, может быть прямой и косвенной. Поскольку в одной рабочей области может быть открыт только один файл БД, то, по существу, ссылка на рабочую область, заданную номером или алиасом, означает ссылку на определенный файл БД.

При открытии файла не в текущей рабочей области командой **USE** нужно указывать в опции **IN** ее номер, так как алиас задавать нельзя, пока файл не открыт. Имя файла можно задавать непосредственно в виде литеральной (буквенно-цифровой) строки до 8 символов или косвенно в виде символьного выражения. Отметим, что, если в качестве выражения применяется переменная, она должна заключаться в круглые скобки, для того чтобы отличить ее от имени, заданного в виде литеральной строки (символьной строки без кавычек). Например, перед открытием файла его имя сначала засылается в переменную:

```
File = "Kadry"
USE (File)
```

Кроме этого для задания имени файла можно использовать функцию макроподстановки **&**, которая означает, что используется не имя переменной, а ее содержимое:

```
USE &File
```

Если файл находится на диске или в директории, отличном от текущего, нужно перед именем файла указывать диск и маршрут по директориям, например:

```
USE C:\DIR1\DAT1\File01
```

Открыть файл можно, также используя опцию **?** в команде **USE**, которая высвечивает список файлов на текущем или другом диске. Каждый файл БД открывается в своей рабочей области.

Особой разновидностью команд являются SET-команды, устанавливающие параметры среды, в которой выполняются обычные команды. Применяются две формы этих команд:

SET <параметр> TO <выражение> и
SET <параметр> ON/OFF

Первая форма устанавливает значение параметра, а вторая включает **(ON)** или выключает **(OFF)** его действие, т.е. устанавливает состояние. Эти команды можно выполнять в программах, в командном режиме, можно также занести их в специальный файл с именем **Config.db**, определяющий параметры среды на весь сеанс работы с пакетом dBASE IV. В этот файл

SET-команды записываются без слова **SET** по следующей форме: **<глагол> = <значение>**. В качестве значений используются слова **ON** (состояние включено) или **OFF** (состояние выключено), а также другие ключевые слова, употребляемые в соответствующей команде. В файл **Config.db** также можно занести специальные команды установки параметров среды dBASE IV, такие как символ приглашения в командный режим, имя внешнего текстового редактора, имя драйвера принтера и др. Более подробно об этих командах изложено в руководстве по dBASE IV фирмы Ashton-Tate. Здесь лишь опишем команду **PRINTER**. Эта команда позволяет установить четыре различных драйвера принтеров и от одного до пяти шрифтов для каждого из принтеров. Для установки драйверов используется команда

PRINTER <номер принтера> = <драйвер>
[NAME <принтер>]
[DEVICE<устройство>]

где номер принтера - это число из диапазона **1...4**, драйвер - имя файла драйвера, принтер - тип принтера, задаваемый символьной строкой, которая появляется в меню **Print** УЦ при выборе требуемого принтера, устройство - тип порта, к которому присоединен принтер. Например, для установки драйвера лазерного принтера Hewlet-Packard LaserJet используется команда

PRINTER 1 = HPLAS100.pr2 NAME "Hewlett-Packard LaserJet" DEVICE COM1

Различные типы шрифтов можно установить командой

PRINTER <номер принтера> FONT <номер шрифта> =
<начальный код>,<конечный код>
[NAME <наименование>]

где номер принтера соответствует номеру, назначенному в предыдущей команде: номер шрифта - это число из интервала **1...5**, назначенное заданному шрифту; начальный и конечный коды - это управляющие символы, первый из которых включает, а второй выключает заданный шрифт (эти коды специфичны для каждого шрифта), наименование представляет собой символьную строку для обозначения заданного шрифта, которая появляется в меню **Print** УЦ. Например, для установки шрифта **Roman-8** в качестве первого шрифта для принтера LaserJet используется команда

PRINTER 1 FONT 1 = {esc}(8U.{esc}(#@,NAME = "Roman-8"

Кроме **SET**-команд и специальных команд установки среды dBASE IV в файле **Config.db** можно назначить размеры блоков оперативной памяти, занимаемой переменными и элементами массивов, запрограммировать функциональные клавиши и

установить цвет различных областей экрана. Более подробные сведения об этих параметрах приводятся при описании соответствующих команд в разд. 3.2 и 3.7.

Кроме команд в языке dBASE IV используются функции, которые позволяют обрабатывать данные (в том числе, выполнять вычисления), а также определять состояния, например, находится ли указатель записей в конце файла: **? EOF()**. После имени функции обычно стоят круглые скобки (кроме функции макроподстановки **&**), в которых могут помещаться аргументы. Функции нельзя использовать самостоятельно, а только вместе с какой-либо командой. Уже упоминавшаяся функция макроподстановки **&** применяется к строковым переменным и означает, что команда оперирует с содержимым переменной, а не с ее именем. Например, в переменную сначала заносится имя поля файла БД, а затем это поле выводится на экран:

```
STORE "Fam" TO Pole
LIST &Pole
```

Последняя команда эквивалентна команде **LIST Fam**. Функция макроподстановки позволяет модифицировать действие команды программно или с клавиатуры и затем повторно ее выполнять в модифицированном виде, не меняя самой записи команды. Можно также полностью заменять в переменной целую команду, что дает возможность программно или с клавиатуры модифицировать часть или всю программу (т.е. при выполнении программы модифицировать самое себя):

```
Comanda = [USE File1]
&Comanda
Comanda = [CLOSE File1]
```

Следует отметить, что псевдокомпилятор dBASE IV при компилировании программ оставляет некомпилироваными команды, в которых используются функции макроподстановки, и возвращается к их компиляции уже в ходе выполнения программы, когда определено конкретное значение переменной с макроподстановкой. Это значительно замедляет процесс выполнения программ. Выше упоминалось, что в командах, включающих имена файлов, можно вместо функции макроподстановки использовать косвенную ссылку на имя файла в виде переменной (заключаемую в скобки), в которую предварительно вводится его имя. При таком способе задания переменного имени файла команда выполняется значительно быстрее.

Кроме системных функций в dBASE IV можно использовать функции, созданные самим пользователем (более подробно см. команду **FUNCTION** в разд. 3.6).

Системные переменные в dBASE IV служат для управления выводом данных на печать и экран, а также позволяют установить (возвратить) состояние параметров этого вывода, что

делает их сходными по своим действиям с **SET**-командами и функциями.

Режим ввода и исполнения команд с клавиатуры отмечается точкой в левом нижнем углу экрана. Выполнение набранной команды осуществляется клавишей **Enter**. Командная строка в этом режиме может быть длиной до **254** символов. Для набора команд, не умещающихся в эту длину, можно использовать режим ввода в окно, который устанавливается клавишами **Ctrl-Home**. В этом случае можно набирать команды длиной до **1024** символа. Команды и некоторые ключевые слова можно записывать не полностью, а первыми четырьмя буквами, например, вместо **MODIFY COMMAND** писать **MODI COMM** и употреблять как прописные, так и строчные буквы, а также их комбинации. Между словами в командной строке допускается любое количество пробелов. Режим ввода команд с клавиатуры имеет свой буфер, называемый **HISTORY**, в котором сохраняются введенные команды. Это позволяет вновь высвечивать на экране ранее введенные команды (навигационными клавишами **Вверх** и **Вниз**), редактировать и выполнять их повторно. По умолчанию буфер позволяет хранить до **20** команд. Его размер можно увеличить до **16000** командных строк с помощью команды **SET HISTORY**. Команды **DISPLAY/LIST HISTORY** позволяют высветить на экране хранящиеся в буфере команды.

3.2. РАЗРАБОТКА БАЗ ДАННЫХ

Создание файла БД осуществляется командой **CREATE <имя файла> STRUCTURE.** Этот процесс был подробно описан в гл. 2 при изложении функций УЦ. Для модификации созданной структуры файла используется команда **MODIFY STRUCTURE.** Эта команда перед модификацией структуры делает копию файла БД с расширением **.bak** и файла Memo полей с расширением **.tbk**. После завершения модификации содержимое одноименных полей из скопированных файлов вводится в модифицированный файл. Следует помнить, что при модификации структуры требуется значительный дополнительный объем дискового пространства. Как уже упоминалось в гл. 2, в функциях УЦ реализованы не все команды языка dBASE IV. В частности, при создании структур файлов БД командный язык предоставляет значительно большие возможности. Например, можно создать копию структуры файла БД (пустой файл БД) с помощью команды **COPY STRUCTURE TO <имя файла> [FIELDS <список полей>]**. При этом можно копировать не все поля, а те, которые указаны в списке.

Пример:

```
USE Fio
COPY STRUCTURE TO Fio1
```

Команда **COPY TO <имя файла> STRUCTURE EXTENDED** позволяет еще больше расширить возможности по созданию структур файлов БД. Данная команда создает файл БД с полями: **FIELD_NAME** (имя поля), **FIELD_TYPE** (тип поля), **FIELD_LEN** (длина поля), **FIELD_DEC** (число десятичных позиций) и **FIELD_IDX** (признак индексирования поля), в которые заносятся параметры структуры текущего файла БД. В этом файле можно заменить значения полей на новые и затем создать структуру файла, соответствующего новым значениям полей, командой **CREATE <имя файла> FROM <имя файла структуры БД>**. Таким способом можно автоматически создавать новые структуры файлов с помощью разработанной для этой цели процедуры. В этой процедуре может понадобиться определить имя поля по его порядковому номеру в структуре файла. Для этого служит функция **FIELD(<выр.N>[,<алиас>])**, которая возвращает имя поля файла БД в соответствии с этим номером в текущей рабочей области или заданной алиасом.

Следует отметить, что файл, созданный командой **COPY TO <имя файла> STRUCTURE EXTENDED** в dBASE III+, не совместим с файлом dBASE IV, так как последний имеет дополнительное поле **FIELD_IDX**. Это поле используется для создания связанного множественного индексного файла.

Как уже упоминалось, в dBASE IV в отличие от dBASE III употребляется новый тип индексного файла - множественный индексный файл с расширением **.mdx**. Этот файл представляет собой собранные вместе обычные индексные файлы, используемые в dBASE III. Индексные файлы в составе множественного индексного файла называются указателями индексов (**TAG**). В dBASE IV создаются два вида множественных индексных файлов - связанный и автономный, которые имеют одинаковое расширение. Отличие их состоит в том, что связанный множественный индексный файл может создаваться автоматически и его указатели являются простыми, т.е. представляют собой поля файла БД, которые указаны для индексирования в поле **FIELD_IDX** файла структуры файла БД (при автоматическом создании). Этому индексному файлу присваивается имя файла БД, и он открывается автоматически вместе с файлом БД. Автономный множественный индексный файл или просто множественный индексный файл создается командой **INDEX ON** (связанный множественный индексный файл также может создаваться этой командой). Указатели индексов в нем могут быть не только простыми, состоящими из одного поля, но и сложными, состоящими из нескольких полей, включенных в индексное выражение, которое можно сформировать из этих полей. Файлам такого типа нужно присваивать уникальное имя. Можно создать до 10 файлов последнего вида, тогда как

связанный множественный индексный файл может быть только один.

Для совместимости в dBASE IV используются индексные файлы dBASE III (типа **.ndx**), а также предусмотрены команды создания этого типа файлов, копирования указателей множественных индексных файлов в этот тип файла и другие операции, связанные с файлами типа **.ndx**. Для создания индексных файлов типа **.ndx** и **.mdx** используется команда

INDEX ON <индексное выражение>
TO <имя .ndx файла>/
TAG <имя указателя индекса .mdx файла>
[OF <имя .mdx файла>] [UNIQUE] [DESCENDING]

Множественный индексный файл создается при использовании опции TAG и может содержать до 47 указателей индексов, которые определяются заданными индексными выражениями. Если индексное выражение включает несколько типов полей, то они должны быть приведены к одному типу. Для этого следует использовать специальные функции, например **STR()**, **SUBSTR()**, **CTOD()**, **DTOC()**, **DTOS()**, **YEAR()**, **MONTH()**, **DAY()** и **VAL()** (см. раздел 3.7). Имена указателей индексов записываются так же, как и имена обычных переменных: содержат до 10 символов (включая буквы, цифры и символы подчеркивания) и должны начинаться с буквы. Для того чтобы определить, что в символьной переменной содержится имя, т.е. значение начинается с буквы, можно использовать функцию **ISALPHA(<выр.C>)**, которая возвращает логическое значение .T., если символьное выражение начинается с буквы.

Если в опции **OF** не задано имя **.mdx** файла, то создается связанный множественный индексный файл. Оглавление файла БД содержит метку, указывающую на наличие связанного множественного индексного файла. Для создания индексного файла типа **.ndx** следует использовать опцию **TO** <имя.**ndx** файла>.

При работе с индексированным файлом БД только один индексный файл типа **.ndx** или указатель множественного индексного файла типа **.mdx** является управляющим, т.е. определяет расположение записей при выводе (напомним, что физическое расположение записей при индексировании не меняется). Управляющим индексом по умолчанию становится вновь созданный индекс. Установка нового управляющего индекса выполняется командами **SET INDEX** или **SET ORDER**.

Для того чтобы индексные файлы автоматически перестраивались при обновлении, удалении или добавлении записей, нужно, чтобы они были открыты. В этом отношении преимуществом обладает связанный множественный индексный файл, который открывается автоматически вместе с файлом БД. Если изменение файла БД выполнялось при закрытых индексных файлах, то

перестроение последних в соответствии с новым состоянием файла БД можно выполнить командой **REINDEX**.

Опция **UNIQUE** позволяет включать в индекс только первую запись с повторяющимся значением индексного выражения. Более подробно описание этой опции приведено в команде **SET UNIQUE**. Опция **DESCENDING** выстраивает указатели **.mdx** файла в уменьшающемся порядке (по умолчанию принят увеличивающийся порядок).

Пример:

```
USE Zakupki
INDEX ON Date TAG Date DESCENDING
LIST
```

Индексные файлы и поля (файлы) **Memo** записываются на диск блоками фиксированной длины. Длина блока определяет скорость манипуляции этими данными. При увеличении блоков имеется тенденция к ускорению обработки их содержимого и одновременно замедлению процесса ввода-вывода. При уменьшении размера блока, наоборот, замедляется обработка содержимого, но в то же время ускоряется ввод-вывод. По умолчанию длина блока равна 512 байт. Команда **SET BLOCKSIZE TO <выр.N>** устанавливает размер блоков полей **Memo** и множественных индексных файлов (**.mdx**) в единицах, кратных **512** байт. Увеличение размера блока для больших полей **Memo**, например до определенного размера, ускоряет процесс их обработки. Но дальнейшее увеличение размера блока приводит к уменьшению коэффициента заполнения блока информацией и, следовательно, к увеличению объема файла, что замедляет его обработку.

Как уже упоминалось для того, чтобы перестроить индексные файлы в соответствии с выполненными изменениями файла БД, сделанными при закрытых индексных файлах, используется команда **REINDEX**, которая осуществляет переиндексацию всех открытых **.ndx** и **.mdx** файлов текущей рабочей области. Статус **UNIQUE** остается неизменным, если файлы были созданы с установленной командой **SET UNIQUE ON** или опцией **UNIQUE** команды **INDEX**, независимо от текущей установки команды **SET UNIQUE**.

Команда **SET UNIQUE ON** действует аналогично опции **UNIQUE** и устанавливает состояние **UNIQUE** - режим включения в индексные файлы (**.ndx** и **.mdx**) только первых записей с повторяющимся значением индексного выражения (эта команда должна быть выполнена до команды **INDEX**).

Команда **SET UNIQUE OFF** (устанавливается по умолчанию) отменяет данный режим. При выполнении команды **REINDEX** над индексом, который был создан в состоянии **UNIQUE**, он сохраняет свой статус **UNIQUE** независимо от текущего значения параметра **SET UNIQUE** (**ON** или **OFF**) в ходе переиндексации.

Обновление и добавление новых записей в файл с открытым индексом не изменяет его статус **UNIQUE**. Это означает, что если, например, с помощью команды **APPEND** добавляется запись, значения полей которой, входящих в индексное выражение, дублируют содержащиеся в индексном файле, то эта запись не включается в индексный файл, но добавляется в файл БД. Подобным образом, если при обновлении записи формируются значения полей, совпадающие с уже существующими, и индексный файл находится в состоянии **UNIQUE**, то эта запись удаляется из индексного файла. Для того чтобы вернуть индексный файл, переведенный в состояние **UNIQUE**, в исходное (не **UNIQUE**) состояние, нужно перестроить индекс с помощью команды **INDEX ON** при состоянии **SET UNIQUE OFF**.

Если при обновлении файла, находящегося в состоянии **UNIQUE**, изменяется запись, и есть скрытые дублирующие записи по проиндексированным полям, то эти записи не вводятся в индексный файл до тех пор, пока не будет выполнена команда **REINDEX**.

Как уже упоминалось, для открытия индексных файлов используется команда

```
SET INDEX TO [/?<список имен файлов>
             [ORDER [TAG] <имя .ndx файла>/
             <имя указателя индекса>
             [OF <имя .mdx файла>]]]
```

Эта команда открывает индексные файлы типа **.ndx** и **.mdx** и может задавать для активного файла БД управляющий индексный файл или указатель индекса множественного индексного файла. Если не указано расширение индексного файла, то сначала выполняется попытка открыть **.mdx** файл, а затем **.ndx** файл.

При выполнении команды **SET INDEX TO** перед открытием индексных файлов закрываются все открытые индексные файлы в текущей рабочей области, кроме связанного файла **.mdx**, затем открываются индексные файлы, указанные в списке имен файлов. При этом указатель записи файла БД устанавливается на первую запись в соответствии с главным (управляющим) индексом.

Команда **SET INDEX TO** без опций закрывает все открытые файлы **.ndx** в текущей рабочей области, а файлы **.mdx** оставляет открытыми. Эти же действия выполняет команда **CLOSE INDEX**.

Индексный файл, указанный в опции **ORDER**, устанавливается как главный или управляющий индекс. Этот индекс может быть файлом типа **.ndx** или указателем множественного индексного файла типа **.mdx**. В последнем случае следует задать имя файла в опции **OF** (для связанного файла **.mdx** это не требуется). Если открыт только один индексный файл, то он автоматически будет главным. В этом случае опция **ORDER** не требуется. При

выполнении команды **SET INDEX TO** указатель файла БД устанавливается на первую запись в соответствии с главным индексом.

При восстановлении естественного порядка обработки файла БД командами **SET ORDER TO 0** или **SET ORDER TO** без опций, все индексы продолжают обновляться. Если естественный порядок обработки файла БД восстанавливается командой **SET INDEX TO** без опций, то только один связанный файл **.mdx** будет по прежнему обновляться. Поэтому при восстановлении индексных файлов командой **REINDEX** можно открыть все индексные файлы, кроме связанного файла **.mdx**.

При выполнении команды **APPEND** с индексированным файлом БД новые записи добавляются в физический конец этого файла. После завершения ввода данных в новую запись индексное выражение этой записи включается в главный индекс и запись становится доступной в соответствии с порядком, определенным этим индексом. При этом обновляются все индексы, включая и все указатели внутри открытого файла **.mdx**. Команда **INSERT** для индексированного файла БД выполянется аналогично команде **APPEND**. Одновременно могут быть открыты все указатели индексов одного файла **.mdx (47)** и до **10 .ndx** и связанных файлов **.mdx**. Если открыт файл **.mdx**, то все указатели в нем также открываются.

Примеры:

```
USE Fio
SET INDEX TO Fam                && Открывает индексный файл Fam.ndx
SET INDEX TO Fio ORDER Fam      && Открывает связанный .mdx файл и
*                                  устанавливает в качестве
*                                  главного указатель Fam
```

Используя команду **SET ORDER**, можно переключить установку главного индекса, не закрывая файлы **.ndx** и **.mdx**:

SET ORDER TO
SET ORDER TO [<выр.N>]
SET ORDER TO [TAG] <имя файла>/
<указатель .mdx файла>
[OF <имя .mdx файла>]

Данная команда позволяет установить любой открытый файл **.ndx** или указатель **.mdx** файла в качестве главного индекса или отменить управление файла БД всеми открытыми индексами, не закрывая их. Это дает возможность сэкономить время, затрачиваемое на открытие и закрытие индексных файлов, если требуется переназначить главный индексный файл, поскольку такие команды не закрывают и не открывают индексные файлы и не перемещают указатель записей. Первая форма команды **SET ORDER TO** без опций отменяет индексный режим обработки, т.е. восстанав-

ливает обработку записей файла БД в соответствии с их физическим расположением и закрывает все индексные файлы.

Для установки в качестве главного только файла **.ndx** используется вторая форма команды **SET ORDER TO <выр.N>** (при этом файлы **.mdx** должны быть закрыты). Числовое выражение содержит числа от **0** до **10**, соответствующие номеру открытого файла **.ndx**, в списке, установленном командами **SET INDEX TO** или **USE**. Эта команда обеспечивает использование индексных файлов типа **.ndx**, созданных в dBASE III+, хотя их можно также создавать и в dBASE IV.

Если одновременно открыты файлы **.mdx** и **.ndx**, то для установки главного индекса используется третья форма команды. В этом случае для задания имен индексных файлов в отличие от второй формы нужно указывать не их порядковые номера, а имена, как непосредственно, так и косвенно, в виде содержимого переменных. При установке в качестве главного указателя множественного индексного файла, не являющегося связанным, следует задать в опции **OF** имя этого файла. Все открытые индексные файлы обновляются, если выполняются изменения значений индексированных полей, добавление или удаление записей.

Команда **SET ORDER TO 0** отменяет главный индекс, но оставляет индексные файлы типа **.ndx** открытым для обновления при изменении файла БД. Для того чтобы определить текущий главный индекс, используется функция **ORDER([<алиас>])**, которая возвращает имя управляющего файла **.ndx** или имя управляющего указателя индекса файла **.mdx** (прописными буквами без расширения). Если алиас не указан, то предполагается текущая рабочая область. Команды, не использующие индексные файлы, например **LOCATE**, действуют быстрее, если все индексы закрыты. В ряде случаев возникает необходимость копирования указателей множественного индексного файла типа **.mdx** в индексные файлы типа **.ndx**, например, в целях использования их в пакете dBASE III. Для этого служит команда

COPY TAG <имя указателя индекса .mdx файла>
[OF имя .mdx файла] TO <имя .ndx файла>

которая создает новый индексный файл типа **.ndx** посредством копирования указателя файла **.mdx**. При копировании файл БД должен быть активным, так как команда создает файл **.ndx** из выражения, содержащегося в указателе файла **.mdx**, который должен быть открытым. Открывается же он только после открытия связанного с ним файла БД. Одна операция копирует только один указатель.

Пример:

```
USE Sklad
COPY TAG Detnomer TO Nazvanie.ndx
```

```
SET INDEX TO Nazvanie
DISPLAY STATUS
```

Для уничтожения ненужных указателей в множественном индексном файле используется команда

DELETE TAG <имя указателя 1>
[OF <имя .mdx файла>]/
<имя .ndx файла 1> [,<имя указателя 2>
[OF <имя .mdx файла>]/
<имя .ndx файла 2>...]

Удаление указателя индекса означает удаление его из индексного множественного файла и освобождение занимаемой памяти. Перед выполнением команды **DELETE TAG** множественный индексный файл должен быть открытым, а удаляемый указатель индекса не должен быть управляющим. Если удаляются все указатели индексов, то одновременно удаляется и весь файл **.mdx**. При удалении связанного файла **.mdx** посредством удаления всех его указателей в оглавлении файла БД метка о наличии связанного файла **.mdx** также удаляется. Если каталог открыт, то в нем отражаются соответствующие изменения.

Опция **OF** позволяет задать имя файла **.mdx**, который содержит удаляемый указатель индекса (для связанного файла **.mdx** эта опция не задается). Например, если два открытых файла **.mdx** содержат указатели индекса с одним именем, то для удаления одного из них требуется указать имя файла **.mdx**.

Вместо удаления указателей файлов **.mdx** данная команда позволяет закрыть индексные файлы типа **.ndx**. В этом случае используется альтернативная опция (в синтаксисе команды отделена косой чертой), в которой задается имя файла **.ndx**. Этот альтернативный вариант команды полезен с целью экономии времени в тех случаях, когда требуется закрыть только несколько файлов **.ndx**, а остальные оставить открытыми. Такой результат можно также получить с помощью команды **SET INDEX**. Однако при применении последней команды все открытые файлы сначала закрываются, а затем вновь открываются заданные, на что тратится больше времени.

Пример:

```
DELETE TAG Fam OF Fio
```

При переходе с пакета dBASE III на dBASE IV может возникнуть потребность в переводе индексных файлов типа **.ndx** в файлы типа **.mdx**. Для этого служит команда

COPY INDEXES <список .ndx файлов>
[TO <имя .mdx файла>]

которая копирует список файлов **.ndx** в указатели файлов в одном множественном индексном файле. Если не указано имя файла **.mdx**, то указатели записываются в связанный файл **.mdx**.

Если же этот файл не существует, то он создается (по определению ему присваивается имя активного файла БД). Одной командой можно копировать до 47 индексных файлов типа **.ndx**.

При работе в среде ЛВС перед выполнением команд **REINDEX**, **COPY TAG**, **DELETE TAG** и **COPY INDEXES** файл БД должен быть установлен в исключительное распоряжение одного пользователя (см. разд. 3.8).

Пример:

```
USE Fio INDEX Fam
COPY INDEX Fam
```

При работе с индексированными файлами БД может возникнуть надобность вывести информацию, записанную в созданные ранее индексные файлы, например о составе индексного выражения, имени файлов **.mdx** и **.ndx**, указателях индексов в файле типа **.mdx**, текущем управляющем индексе и т.д. Для этого служат специальные функции. Напомним, что функции можно использовать только в составе команд. В командном режиме для вывода информации на экран часто используется функция ? (см. разд. 3.4). Функция **KEY([<имя .mdx файла>,] <выр.N> [, <алиас>])** возвращает индексное выражение файлов типов **.mdx** и **.ndx**, которое задано числовым выражением, определяющим порядковый номер файла в текущей рабочей области. Для файла **.mdx** нужно задать его имя, а выражение будет определять порядковый номер указателя индекса. Если требуется вывести информацию о файле, находящемся в другой рабочей области, то указывается ее алиас. Для вывода имен файлов типа **.mdx** используется функция **MDX(<выр.N> [,<алиас>])**, которая возвращает имя файла **.mdx**, указанного порядковым номером в списке индексных файлов, установленном командой **SET INDEX TO** в текущей рабочей области или заданной алиасом. При отсутствии **.mdx** файлов возвращается нулевая строка. Функция **NDX(<выр.N>[,<алиас>])** возвращает имя файла **.ndx**, соответствующее заданному числовому выражению, которое представляет собой номер позиции в списке индексных файлов текущей рабочей области или заданной алиасом. Если командами **SET INDEX TO** или **USE INDEX** открыто **N** файлов **.ndx**, то число, полученное из заданного выражения в диапазоне от 1 до N, будет возвращать имена этих **.ndx** файлов, а числа свыше будут соответствовать указателям файлов **.mdx** и каталогизированным файлам **.ndx** (при открытом каталоге).

Функция **TAG([<имя .mdx файла>,] <выр.N> [,<алиас>])** возвращает имя указателя индекса в заданном индексном множественном файле **.mdx** в соответствии с указанным числовым выражением. Если имя файла **.mdx** не указывается, то возвращается имя любого открытого индексного файла из списка файлов в текущей (заданной алиасом) рабочей области в соответствии с числовым выражением. При отсутствии открытых индексов

возвращается нулевая строка. Для вывода имени текущего главного индекса - индексного файла типа **.ndx** или указателя индекса в файле типа **.mdx** - в текущей или заданной алиасом рабочей области используется функция **ORDER([<алиас>])**.

Возвращаемые функциями dBASE IV имена файлов содержат полный маршрут, тогда как эти же функции в dBASE III+ возвращают только имя диска и имя файла.

Команда **SET FULLPATH OFF** (устанавливается по умолчанию) подавляет (**ON** не подавляет) в возвращаемом вместе с именем файла маршруте имена директориев, что обеспечивает совместимость с dBASE III+ программ, написанных в dBASE IV и содержащих функции **MDX()**, **NDX()** и **DBF()**.

Перед выполнением каких-либо операций с ранее созданным файлом его надо открыть. Как уже говорилось, для обеспечения обработки файла БД в оперативной памяти выделяется специальный участок - рабочая область. В каждой рабочей области можно открыть один файл БД. Рабочая область **10** зарезервирована для каталога. Если каталог открыт, то в этой области открывать файл БД нельзя. Если в рабочей области **10** открыт файл БД, а затем открывается каталог, то файл БД автоматически закрывается. При открытии файла БД в области, уже содержащей открытый файл, последний закрывается. В рабочей области вместе с файлом БД также открываются и связанные с ним файлы других типов, например, индексный файл, файл запроса, экранной формы и др. При входе в dBASE IV автоматически активизируется рабочая область **1**. Выбор рабочей области осуществляется командой

SELECT <имя рабочей области/алиас>

Эта команда активизирует рабочую область для открытия нового файла БД или для работы с уже открытым в этой области файлом БД.

Команда **SELECT** также используется для переходов между рабочими областями. Файл БД, открытый в выбранной рабочей области, становится активным. Как уже упоминалось, рабочую область можно выбрать, задав алиас (псевдоним) имени файла, открытого в этой рабочей области. Если при открытии файла не задан алиас, то имя файла (без расширения) по умолчанию принимается в качестве алиаса.

При выборе рабочих областей или алиасов можно прямо задавать не только их имена, но и переменные, содержащие эти имена. Такой способ задания особенно полезен при применении в прикладных программах. В этом случае для выбора различных рабочих областей можно использовать одну команду, в которой изменяется значение переменной. Тогда перед переменной нужно ставить функцию **&**, которая означает, что выбирается не имя переменной, а ее значение. Например, если алиас содержится в символьной переменной, то следует указать:

SELECT &<переменная>

Обычно команды выводят или изменяют значения полей активного файла БД. Тем не менее можно задать поля из других открытых рабочих областей, содержащих связанные файлы, используя следующий синтаксис: **<алиас> -> <имя поля>**. Если предварительно установлены команды **SET FIELDS TO** и **SET FIELDS ON**, то будут доступны поля из файлов в других открытых рабочих областях, определенные в списке первой из этих команд.

В каждой рабочей области поддерживается свой указатель записи. Переходы между рабочими областями не изменяют положения указателей записи. Команды, изменяющие позицию указателя записи, действуют только для активного файла БД, если не установлена связь файлов с помощью команды **SET RELATION TO** или не был открыт файл выборки. Напомним, что выборкой называется файл БД, получаемый в результате отбора данных из нескольких файлов. При этом исходные файлы могут быть вообще не связаны или же связаны логически с помощью команды **SET RELATION**.

Примеры:

```
SELECT 1        && В области 1 открывается файл Fio с алиасом Names
USE Fio ALIAS Names
SELECT 3
USE Finans      && В рабочей области 3 открывается файл Finans
SELECT Names    && Рабочая область выбирается по алиасу
SELECT A        && Выбирается рабочая область 1
```

Для того чтобы определить алиас рабочей области, используется функция **ALIAS([<выр.N>])**, которая возвращает алиас рабочей области, указанной в числовом выражении. Если номер рабочей области не указан, то подразумевается текущая. Установить номер высшей неиспользованной рабочей области можно с помощью функции **SELECT()**.

Пример:

```
Mfile = "Fio"
USE (Mfile) IN SELECT() && Открывает файл БД Fio в высшей
*                          неиспользованной рабочей области
```

Для открытия файла БД используется команда

USE [<имя файла БД>/?] [IN <номер рабочей области>]
[INDEX <список .ndx и .mdx файлов>]
[ORDER [TAG] <имя .ndx файла>/
<указатель .mdx файла>
[OF <имя .mdx файла>]]
[ALIAS<алиас>] [EXCLUSIVE] [NOUPDATE]

которая открывает существующий файл БД и связанные с ним файлы **.mdx** и **.ndx** в текущей рабочей области.

Для открытия в другой рабочей области требуется задать опцию **IN**. Если файл БД содержит поля **Memo**, автоматически открывается соответствующий файл **.dbt**. Одновременно может быть открыто до **94** файлов различных типов, но только **9** из них могут быть файлами БД, каждый в своей рабочей области. Остальные файлы - это связанные с файлами БД файлы индексов, экранных форм, отчетов и т.д., которые открываются в одной рабочей области со своим файлом БД. Напомним, что 10-я рабочая область отводится под файл каталога.

Команда **USE** без параметров закрывает активный файл БД и связанные с ним индексы в текущей рабочей области. Команда **USE IN** закрывает файлы в указанной рабочей области. Если не указывается расширение файла, то предполагается, что это файл БД (имеет расширение **.dbf**), а если не указывается расширение индексных файлов, то для них предполагаются расширения **.mdx** или **.ndx**. При этом сначала осуществляется попытка открыть файл **.mdx**. При отсутствии файла **.mdx** предпринимается попытка открыть файл **.ndx**. После того как открыт файл БД, можно открывать и закрывать индексные файлы с помощью команды **SET INDEX** и закрывать командами **CLOSE INDEX** и **DELETE TAG**. Если имя файла БД еще не записано в каталоге и установлена команда **SET CATALOG ON**, то оно заносится в каталог.

Команда **USE ?** высвечивает список файлов БД в каталоге, если он открыт, или список всех файлов БД в текущем директории при закрытом каталоге. Опция **ALIAS** назначает алиас файла БД, который в дальнейшем может использоваться для ссылки на него. Напомним, что если алиас не задан, то по умолчанию он совпадает с именем файла БД.

Опция **EXCLUSIVE** устанавливает файл БД в исключительное распоряжение одного пользователя при работе в среде ЛВС. Этот файл не может использоваться другими пользователями, пока не будет закрыт командами **CLEAR ALL, CLOSE** или **USE**. В однопользовательском режиме все файлы открываются со статусом **EXCLUSIVE**.

Опция **NOUPDATE** защищает открытый файл БД от всякого изменения (обновления, добавления или удаления записей), т.е. фиксирует режим только чтения.

Для установки управляющего индекса используется опция **ORDER**. Если в списке индексов указан только один индексный файл, то эта опция не требуется. При использовании в качестве управляющего указателя не связанного файла **.mdx** требуется в опции **OF** задать его имя. После открытия файла БД указатель записи устанавливается на первую запись в соответствии с заданным управляющим индексом, который затем определяет перемещение указателя записи при работе с файлом. Если открывается файл БД, не имеющий индексов, указатель устанавливается на физически первую запись в файле.

В гл. 2 подробно описана система каталогизации файлов в пакете dBASE IV. Здесь лишь поясним команды для работы с каталогами. Для создания или открытия существующего каталога используется команда

SET CATALOG TO [<имя файла каталога>]/?

Выполнение этой команды начинается с поиска заданного файла каталога в текущем директории и в директории, в котором размещается dBASE IV. Если файл каталога найден, то он открывается. В противном случае создается новый файл каталога (с расширением **.cat**). Если создаваемый каталог первый, то предварительно формируется главный каталог **Catalog.cat**. Этот каталог автоматически помещается в директорий, в котором находится dBASE IV, и содержит информацию о других каталогах. Его можно удалить из директория dBASE IV или переместить в другой директорий. После удаления из директория dBASE IV главный каталог автоматически появляется в текущем директории. Таким образом, на каждом диске или в каждом директории можно создать свой главный каталог.

Команда **SET CATALOG TO ?** высвечивает список каталогов, находящихся в главном каталоге вместе с их описаниями, которые можно ввести при создании каталога, если установлена команда **SET TITLE ON**. В этом случае при создании нового каталога высвечивается подсказка для ввода описания каталога. Если установлена команда **SET TITLE OFF**, то описание каталога после его создания можно ввести непосредственно в соответствующее поле файла главного каталога, открытого в любой рабочей области, например с помощью команд **EDIT** или **REPLACE**.

Кроме этого способа можно ввести описание каталога через меню УЦ, описанное в гл. 2. Отметим, что для открытия файла каталога в отведенной для него рабочей области 10 используется команда **SET CATALOG TO**. Если каталог открыт и установлена команда **SET CATALOG ON**, то все новые файлы (файлы БД, связанные с ними индексные файлы, а также запросы, отчеты, марки и т.д.), создаваемые командами

COPY STRUCTURE	IMPORT FROM
COPY STRUCTURE	EXTENDED INDEX
CREATE	JOIN
CREATE FROM	SET FILTER TO FILE
CREATE/MODIFY LABEL	SET FORMAT
CREATE/MODIFY QUERY	SET VIEW
CREATE/MODIFY REPORT	SORT
CREATE/MODIFY SCREEN	TOTAL
CREATE/MODIFY VIEW	USE

будут в него заносится. По умолчанию установлено состояние **SET CATALOG OFF**, которое не позволяет каталогизировать файлы, но при выборе каталога командой **SET CATALOG TO**

автоматически устанавливается состояние **SET CATALOG ON**. Если кроме **SET CATALOG ON** установлено еще состояние **SET TITLE ON**, то при создании нового файла указанными выше командами высвечивается приглашение для ввода в каталог его описания. При открытии каталога выполняется сверка его содержимого с перечнем файлов на диске. Если ранее было выполнено удаление файлов при состоянии **SET CATALOG OFF**, то их имена удаляются из каталога.

Закрытие каталога выполняется командой **SET CATALOG TO** без имени файла. Если требуется только прекратить добавление файлов в открытый каталог, следует использовать команду **SET CATALOG OFF**. Возобновление ввода файлов в каталог осуществляется командой **SET CATALOG ON**.

Команда **SET CATALOG OFF** в отличие от команды **SET CATALOG TO** оставляет каталог открытым и позволяет использовать опцию запроса к каталогу ? в других командах dBASE IV. Например, если установлено состояние **SET CATALOG OFF**, то команда **MODIFY REPORT?** высвечивает список имен файлов отчета из открытого каталога. В список будут включены все файлы, связанные с активным файлом БД.

Как уже упоминалось, в состоянии **SET CATALOG OFF** файлы не заносятся в каталог. Например, созданный в этом состоянии индексный файл не помещается в каталог. Если позже устанавливается состояние **SET CATALOG ON** и задается обращение к этому индексному файлу, то в этом случае высвечивается сообщение: **File not cataloged, since SET CATALOG was OF when the active database file was USEd** (файл не каталогизирован, так как было установлено состояние **SET CATALOG OFF** при работе с файлом БД).

Файлы каталогов имеют одинаковую структуру (табл. 3.1). В поле **File_name** записывается имя файла, включая его расширение, а в поле **Alias** (заполняется только для файлов БД) его алиас.

Таблица 3.1 **Структура файлов каталогов**

Field (номер поля)	Field Name (имя поля)	Type (тип)	Width (ширина)	Dec (число десятичных позиций)
1	Path	Character	2	
2	File_name	Character	12	
3	Alias	Character	8	
4	Type	Character	3	
5	Title	Character	80	
6	Code	Numeric	3	
7	Tag	Character	4	
Всего			181	

Все поля файла каталога, за исключением **Title** и **Tag**, заполняются системой автоматически при включении нового файла в каталог. Поле с именем **Path** содержит маршрут по директориям для поиска файла, если он задан при создании файла. Если алиас не задан, то в качестве него используется имя файла БД. В поле **Type** находится расширение имени файла, принятое по умолчанию. Если при создании файла ему присвоено другое расширение, то оно помещается вместе с именем файла в поле **File_name**. Поле **Title** содержит словесное описание файла (до **80** символов), введенное при установленном состоянии **SET TITLE ON**. Изменение описания файла в каталоге осуществляется так же, как и описание каталога в главном каталоге, например с помощью команды **EDIT**.

Пример:

```
USE <имя каталога>.CAT
EDIT
```

Если каталог открыт, то можно просто выбрать рабочую область 10.

Пример:

```
SELECT 10
EDIT
```

В поле **Code** помещается номер, который присваивается каждому открытому файлу БД. Программным файлам присваивается **0**. Файлу БД присваивается номер при его создании. Каждому новому файлу БД присваивается следующий номер. Связанным с файлом БД файлам (индексам, экранным формам, запросам, отчетам, маркам и т.д.) присваивается тот же номер, что и файлу БД. Это поле используется для того, чтобы можно было выбрать все файлы различных типов, связанные с одним файлом БД.

Иногда возникает потребность определить имя открытого файла БД в рабочей области. Этим целям служит функция **DBF(<алиас>)**, которая возвращает имя файла БД, открытого в рабочей области, заданной алиасом.

Для проверки существования указанного файла используется функция **FILE(<выр.C>)**, которая возвращает логическое значение **.T.**, если файл существует. Кроме того, часто применяется функция **LUPDATE([<алиас>])**, возвращающая дату последнего обновления указанного файла БД.

После того как создана структура файла БД, можно вводить данные. Как уже отмечалось в гл. 2, существуют две стандартные экранные формы ввода (и вывода) информации в файлы БД: **Browse** и **Edit**. Первая форма вызывается командой

BROWSE [NOINIT] [NOFOLLOW] [NOAPPEND] [NOMENU]
[NOEDIT] [NODELETE] [NOCLEAR] [COMPRESS]

**[FORMAT] [LOCK <выр.N>] [WIDTH <выр.N>]
[FREEZE <имя поля>] [WINDOW <имя окна>]
[FIELDS <имя поля 1> [/R] [/<ширина колонки>]
/<вычисл. имя поля 1> = <выражение 1>
[,<имя поля 2> [/R] [/<ширина колонки>]
/<вычисл. имя поля 2> = <выражение 2>]...]**

Эта команда имеет собственное меню, которое активизируется клавишей **F10** и предназначено для ввода, добавления и обновления записей в файлы БД. В дальнейшем при описании данной команды под файлом БД будет также подразумеваться и выборка. Напомним, что в форме **Browse** записи файла БД выводятся на экран в табличной форме.

В команде имеется ряд опций, позволяющих устанавливать различные режимы ее использования. В число этих опций входит группа опций, позволяющая назначить режимы работы с полями, и группа опций по размещению полей на экране. В состав первой группы входят следующие опции: **NOEDIT**, позволяющая установить режим только чтения для всех полей, **NOAPPEND**, не разрешающая добавлять новые записи, и **NODELETE**, не разрешающая помечать записи для удаления. Опция **FREEZE** назначает для обновления только одно указанное поле. Другие поля высвечиваются, но остаются недоступными для редактирования. Если для указанного в опции **FREEZE** поля командой **PROTECT** открыто только чтение (см. разд. 2.8 и 3.8), то обновление не разрешается. Опция **FREEZE** без имени поля отменяет ранее назначенную установку.

Поля высвечиваются в том порядке, который определен структурой файла БД или указан в команде опцией **FIELDS**, входящей в состав второй группы. Эта опция позволяет выбрать поле и определить порядок его расположения в таблице **Browse**. Установленные командой **PROTECT** привилегии перекрывают опции команды **BROWSE**. Если доступ к полю закрыт командой **PROTECT**, то это поле не высвечивается. Опция **FIELDS** также позволяет устанавливать для выбранного поля режим только чтения, а также создавать высвечиваемое вычисляемое поле по форме, приведенной в синтаксисе опции. Имя этого поля становится заголовком колонки таблицы **Browse**, в которой высвечивается значение вычисляемого поля. Длина этого поля определяется после вычисления выражения для первой записи.

Опция **/R** устанавливает поле в режим только чтения. Если этот режим также назначен командой **PROTECT**, то, как указано выше, команда имеет приоритет над данной опцией. Вычисляемые поля всегда устанавливаются в режим только чтения. Для определения ширины колонки (в которой высвечивается поле), отличной от установленной по умолчанию, используется опция **<ширина колонки>**, в которой задается число, определяющее новый размер колонки (перед ним ставится знак /). Диапазон

составляет от **4** до **100** для символьных полей и от **8** до **100** для числовых полей и полей типа даты. Для логических и полей **Memo** данная опция не действует. Эта опция также игнорируется, если заданное число больше назначенного опцией **WIDTH** или меньше ширины поля числовых полей и полей даты, определенных в структуре файла БД. Опция **WIDTH** устанавливает верхний предел ширины колонки для всех полей в таблице **BROWSE**. Эта ширина перекрывает ширину поля, назначенную при создании файла БД. Если опции **WIDTH** и **<ширина колонки>** используются одновременно, то принимается наименьшее значение. Данная опция, как и предыдущая, не применяется для полей типа **Memo** и логических полей. Значения числовых полей и полей даты не будут высвечиваться, если значение опции **WIDTH** меньше ширины этих полей, назначенных при определении структуры файла БД. Опция **FORMAT** позволяет разместить поля в таблице **BROWSE** в очередности, определенной экранной формой, которая установлена командой **SET FORMAT TO** и использовать при работе с полями все опции команд **@...GET** этой экранной формы, кроме позиционирования полей (см. разд. 3.4.). Это дает возможность подключать к команде **BROWSE** такие возможности, как проверку вводимых данных на соответствие их допустимому диапазону, высвечивание сообщения внизу экрана при установке курсора на поле, ввод альтернативных значений поля и т.д.

Для того чтобы зафиксировать поля при движении курсора, используется опция **LOCK**. В ней задается количество смежных полей в левой части экрана, которые будут оставаться на экране при движениии курсора вправо по полям таблицы **Browse**, осуществляемом клавишами **Tab**, **Shift-Tab**, **F3** или **F4**. Для отмены фиксации полей задается опция **LOCK** без числового значения или с выражением, равным нулю. Опция **WINDOW** позволяет активизировать окно, которое затем определяет на экране область, используемую для высвечивания таблицы **Browse**. При выходе из команды **BROWSE** окно автоматически убирается, если не задана опция **NOCLEAR**. Опция **NOMENU** удаляет меню команды, а **NOCLEAR** оставляет таблицу **Browse** на экране после выхода из сеанса работы с командой. Чтобы разместить на экране на две записи больше, чем обычно (на 25-строчном экране), применяется опция **COMPRESS**, которая сжимает заголовки полей.

Опция **NOINIT** позволяет переносить в данную команду установки всех опций из предыдущей команды **BROWSE**. После обновления индексированного поля записи она автоматически перемещается в новое место в соответствии с новым значением поля. Иногда требуется оставить запись после ее обновления на старом месте. Для этого используется опция **NOFOLLOW**.

Пример:

```
USE Transact && Отбирает записи, обновленные в марте
SET FILTER TO MONTH(Date_trans) = 3
GO TOP
DEFINE WINDOW Win1 FROM 2,10 TO 9,79
BROWSE NOAPPEND NOEDIT NODELETE NOCLEAR NOMENU;
COMPRESS WINDOW Win1
DO Next_prg
```

Выше в команде **BROWSE** была упомянута опция **WINDOW**, применяемая для активизации ранее созданного окна, в которое направляются выводимые командой **BROWSE** данные. Для создания окна используется команда

DEFINE WINDOW <имя окна>
FROM <строка 1>,<колонка 1>
TO <строка 1>,<колонка 1>
[DOUBLE/PANEL/NONE/<символы рамки>]
[COLOR [<стандартная область>]
[,<нестандартная область>] [,<рамка>]]

которая устанавливает его размеры, местоположение на экране, рамку и цвет. Опции **FROM** и **TO** определяют координаты левого верхнего и правого нижнего углов окна (номера строк и колонок), а опция **DOUBLE/PANEL/NONE/<символы рамки>** - тип рамки: двойная линия, полоса, без рамки или рамка из заданных символов. По умолчанию высвечивается одинарная линия рамки. Опция **COLOR** позволяет установить цвета текста, фона и рамки. Для изменения параметров рамки и цвета используются специальные команды. Например, команда

SET BORDER TO [SINGLE/DOUBLE/PANEL/NONE/
<символы определения рамки>]

изменяет параметры рамок окон и меню (см. разд. 3.6). Могут быть назначены различные символы для определения сторон рамки, которые записываются в виде

[<1>[,[<2>[,[<3>[,[<4>[,[<5>[,[<6>[,[<7>[,[<8>]]]]]]]]
где цифры 1-8 обозначают десятичные коды ASCII любого символа, а также стороны и углы рамки (рис. 3.1).

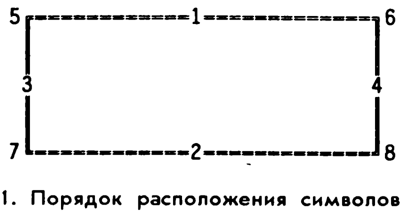


Рис. 3.1. Порядок расположения символов рамки

Команда **SET BORDER TO** без опций восстанавливает символы рамки, принятые по умолчанию (одинарную линию). Опции **SINGLE, DOUBLE, PANEL, NONE** задают соответственно одинарную, двойную, сплошную линии и без рамки.

Пример:

```
SET BORDER TO 205,205,179,179,213,184,212,190
```

Цвета и атрибуты экрана при работе с цветным и монохромным видеомониторами можно установить с помощью команды

```
SET COLOR ON/OFF
SET COLOR TO [[<стандартная область>]
             [,[<нестандартная область>]
             [,[<рамка>] [,[<фон>]]]]]
SET COLOR OF NORMAL/MESSAGES/TITLES
             /BOXS/HIGHLIGHT
             /INFORMATION/FIELDS
             TO [<атрибуты цвета>]
```

Первый вариант команды переключает на использование цветного (**ON**) или монохромного (**OFF**) видеомонитора. По умолчанию принимается значение, заданное при запуске dBASE.

Второй вариант команды позволяет установить цвет текста и фона для стандартной и нестандартной областей экрана, а также цвет рамки. В качестве примера элементов стандартной и нестандартной областей экрана можно привести поля **SAY** и **GET** в команде **@...SAY...GET** (см. разд. 3.4). Различный цвет этих элементов делает более удобной работу с экранной формой ввода и вывода данных. Полный состав элементов стандартной и нестандартной областей экрана приводится ниже. Цвет текста и фона в этих областях можно задать кодами цвета, разделенными символом (/).

Опция **<фон>** используется для адаптеров видеомонитора, на которых можно задать только один цвет фона стандартной и нестандартной областей экрана. При этом в каждой области сначала задается цвет текста, затем цвет рамки и фона.

Команда **SET COLOR TO** без опций восстанавливает принятые по умолчанию в dBASE IV, а не установленные в файле **Config.db** цвета экрана. При старте dBASE IV установки цвета, заданные в файле **Config.db**, перекрывают установки, заданные по умолчанию. Третий вариант команды позволяет установить цвет групп элементов экрана, приведенных ниже. Кроме данной команды цвета можно установить с помощью опции **DISPLAY** команды **SET** (см. разд. 3.4). Буквенные коды цвета, который можно назначить элементам экрана, приведены в табл. 3.2.

После буквы, обозначающей цвет, можно указать символ * для мерцания или знак + для повышенной яркости (кроме коричневого, желтого и серого цвета). Если задана повышенная яркость, цвет воспроизводится более интенсивно.

Таблица 3.2. **Коды цвета экрана**

Цвет	Код
Черный	N или пробел
Синий	B
Зеленый	G
Голубой	BG
Пустой	X
Серый	N+
Красный	R
Фиолетовый	RB
Коричневый	GR
Желтый	GR+
Белый	W

Например, команда **SET COLOR TO G/B,RB/N,BG** устанавливает зеленый текст на синем фоне в стандарной области, фиолетовый текст на черном фоне в нестандартной области и голубой цвет для рамки, а команда **SET COLOR TO B,G,RB,R** высвечивает текст стандартной области синим цветом на красном фоне, текст нестандартной области зеленым цветом на красном фоне и задает фиолетовый цвет рамки. В последней команде можно устанавливать (изменять) не все, а отдельные цвета компонентов экрана, пропуская другие (разделители - запятые - должны оставаться). Например, если требуется установить зеленый цвет для рамки, не затрагивая остальных компонентов, следует указать: **SET COLOR TO,,G**. При установке значений пар текст/фон можно пропустить код, задающий цвет текста или фона. Если в паре текст/фон пропущен один из кодов, для этого параметра будет установлен черный цвет. Если высвечивание вводимых символов в заданной области экрана нежелательно (например, при вводе с клавиатуры пароля), следует задать для текста или фона этой области "**пустой**" цвет (**X**).

Можно сохранить значения цвета элементов экрана в файле **Config.db**, используя следующий синтаксис: **COLOR = <значения>**. Параметры цвета в программах, созданных для цветного видеомонитора, имеют непредсказуемый эффект при переносе их на ПК с монохромным видеомонитором.

При работе с монохромным видеомонитором можно задать код **U** для подчеркивания и код **I** для инвертированного изображения вместо цвета с целью выделения стандартной и нестандартной области экрана. Вообще при переходе на другой монитор целесообразно убедиться в правильности реакции его адаптера на параметры цвета или коды **U** или **I**.

Для определения типа адаптера монитора можно использовать функцию **ISCOLOR()**, которая возвращает логическое значение **.T.**, если установлен цветной графический адаптер. Эта

функция позволяет программно устанавливать коды цвета в зависимости от типа адаптера монитора.

Пример:

```
IF ISCOLOR()
SET COLOR TO GR/B,W/R,GR
ELSE
SET COLOR TO W+
ENDIF
```

При использовании третьего варианта команды можно установить цвета отдельных групп элементов, высвечиваемых на экране. В состав стандартной области экрана входят группы с именами **NORMAL**, **MESSAGES**, **TITLES**, **BOX**, а в нестандартную область - группы с именами **HIGHLIGTH**, **INFORMATION**, **FIELDS**. В группу **NORMAL** входят: информация, выводимая опцией **SAY** команды **@...SAY...GET**, невыбранные поля таблицы **Browse**, экран разработки запросов, отчетов и т.д., невыбранный текст на экране разработки, невыбранные нерасцвеченные шаблоны полей, рамки окон и статических полей **Memo**, условия в шаблоне файла и в окне, вычисляемые поля, невыбранные нерасцвеченные рамки в текстовом редакторе, нерасцвеченные рамки, рисуемые командой **@...TO**, текст, введенный пользователем в генератор прикладных программ. В группу **MESSAGES** входят: информация, высвечиваемая в строке сообщений, в навигационной строке, невыбранные позиции меню и списков, внутренняя часть окна сообщений об ошибках, справочной информации и подсказок, имена файлов в окнах, жирный текст в окне справочной информации, невыбранные позиции окна подсказок, сообщений об ошибках и справочной информации. В группу **TITLES** входят: заголовки списков, окна справочной информации и УЦ, заголовок каталога и его имя, имена файлов и заголовок их описания в УЦ, имена полей и сетка таблицы **Browse**, заголовки, номера полей в таблице разработки структуры файла БД и линии этой таблицы, имя директория в верхней части окна имен файлов, имена файлов и данных для их сортировки, невысвеченные полосы экрана разработки отчета, имена файлов и их полей в шаблонах файла, шаблоны вывода и вычисляемых полей, а также линии этих шаблонов на экране разработки запросов, рамка и заголовок окна задаваемого условия отбора записей, линия разметки экрана в редакторе текста, участки текста, высвечиваемые опциями меню **Style**, подчеркнутый текст в окнах справочной информации. В группу **BOX** входят: рамки меню и информация о файле при его сортировке, рамки окна имен файлов и списков, сообщений и окна разработки марок, а также невыбранного компонента на экране генератора прикладных программ. В группу **HIGHLIGHT** входят: подсвеченные позиции меню и списков, высвеченный текст на экранах разработки УЦ, выбранное окно, имена файлов и их описание в УЦ, курсор,

движущийся по линии разметки экрана, имена полос и высвеченные полосы экрана разработки отчета, рамки и внутренняя часть информационных окон, текущее поле в генераторе прикладных программ, рамки и поля на экранах разработки, на которых стоит курсор.

В группу **INFORMATION** входят: часы, рамки окна ошибок и справочной информации и выбранные позиции в них, строка состояния на экране. В группу **FIELDS** входят: область ввода данных в сообщениях, выбранные поля в таблице **BROWSE**, редактируемые поля команды **@...SAY...GET**, выбранное поле в таблице создания файла БД, редактируемые шаблоны полей на экранах разработки и невыбранные поля в генераторе прикладных программ.

Пример:

```
SET COLOR OF INFORMATION TO B+/G
```

Если не требуется устанавливать различные цвета стандартной и нестандартной областей экрана, можно использовать команду **SET INTENSITY OFF** (по умолчанию **ON** - цвета этих областей различаются). При этом, в частности, поля **SAY** и **GET** переменных команды **@...SAY...GET** рассматриваются как элементы стандартной области экрана.

Чтобы созданные окна можно было многократно высвечивать, их нужно записать на диск командой

SAVE WINDOW <список имен окон>
/ALL TO <имя файла>

По умолчанию имя файла окон имеет расширение **.win**. Можно задать другое расширение этого файла, указывая затем его при вызове окон. Опция **ALL TO** сохраняет в заданном файле все находящиеся в памяти окна. Для вызова окон с диска в память используется команда

RESTORE WINDOW <список имен окон>
/ALL FROM <имя файла>

Этой командой вызываются с диска (без их высвечивания) все сохраненные окна (опция **ALL**) и выборочно в соответствии с заданным списком, в любом порядке, независимо от последовательности их сохранения командой **SAVE WINDOW**. Вызываемое с диска окно перекрывает окно с тем же именем, находящемся в памяти. После вызова в память окон командой

ACTIVATE WINDOW <список имен окон>/ALL

можно активизировать их - высветить на экране заданные в списке окна и направить поток информации в активное окно (последнее в списке). Опция **ALL** высвечивает все окна, находящиеся в памяти, в последовательности их определения, последнее из которых будет активным. Если требуется снова

выводить информацию на весь экран, используется команда **ACTIVATE SCREEN**, которая восстанавливает доступ к полному экрану в отличие от доступа к части экрана, определенного последним активизированным окном. Это окно остается на экране, но перестает быть активным. Данную команду можно применять, например, для помещения текста вне окна в качестве справочной информации, после чего надо опять активизировать окно.

Следует отметить, что при полноэкранном доступе вся информация, ранее выводимая в окно, будет теперь выводится на весь экран, в том числе и в область окна. Это приведет к перекрытию информации, оставшейся в окне или вызовет его перемещение за пределы экрана. Для полной очистки экрана можно использовать команду **CLEAR**. После повторной активизации окно перекрывает информацию, которая попадает в область его размещения.

MOVE WINDOW <имя окна> TO <строка, колонка>/BY <приращение строки, приращение колонки>

Можно задавать координаты нового положения непосредственно или в виде приращения к текущей позиции.

В поля **Memo** файла БД обычно записывается текст (например, пояснения к основным полям текущей записи этого файла), который нередко требуется вывести на экран в отдельное окно. Для этого можно использовать команду **SET WINDOW OF MEMO TO <имя окна>**. В этом окне поля **Memo** можно высвечивать командами **APPEND**, **BROWSE**, **CHANGE**, **EDIT** или **@...SAY...GET**. Опция **WINDOW**, которая задана в командах **@...GET** в экранной форме или в форме **BROWSE**, перекрывает окно, установленное данной командой (см. раздел 3.4).

Альтернативой команды **BROWSE** является команда **EDIT** (переключение с формы **Browse** на форму **Edit** и обратно осуществляется клавишей **F2**):

EDIT [NOINIT] [NOFOLLOW] [NOAPPEND] [NOMENU]
[NOEDIT] [NODELETE] [NOCLEAR] [<номер записи>]
[FIELDS <список полей>] [<диапазон>]
[FOR <условие>] [WHILE <условие>]

Эта команда выводит на экран по одной записи файла БД, поля которой размещаются по вертикали. Если все поля не помещаются на одной стандартной странице экрана, то они продолжаются на следующих страницах, переход к которым осуществляется навигационными клавишами. Команда **EDIT** также служит для вывода данных на экране в ранее разработанной форме (см. разд. 2.5). Для этого надо предварительно выполнить команду **SET FORMAT**, чтобы установить файл экранной формы.

Описание опций команды **EDIT**, совпадающих с опциями команды **BROWSE**, дано выше в описании этой команды. Опция **<номер записи>** позволяет после старта команды высветить сразу запись с заданным номером. Аналогичный результат можно получить, если использовать ключевое слово **RECORD N** опции **<диапазон>**, где **N** - номер записи. Различие между ними состоит в том, что в первом случае после обработки высвеченной записи можно передвигаться к другим записям файла БД, а во втором случае это сделать нельзя. Опции **<диапазон>**, **FOR** и **WHILE** определяют диапазон выводимых записей, условия отбора и завершения процесса вывода записей. Напомним, что в качестве значений опции **<диапазон>** может использоваться упомянутое ключевое слово **RECORD <N>** (выводится запись с заданным номером), а также **NEXT <N>** (выводятся N записей, начиная с текущей), **ALL** (выводятся все записи) и **REST** (выводятся записи, начиная с текущей до конца файла). Опция **FOR** позволяет высвечивать только те записи, которые удовлетворяют заданному условию. Опция **WHILE** задает условие, по которому завершается вывод записей.

Кроме команды **EDIT** в командном языке dBASE IV есть команда **CHANGE**, также предназначенная для обновления данных в файле БД, синтаксис и использование которой полностью совпадают с командой **EDIT**. Команды **BROWSE** и **EDIT** имеют свое меню, которое описано в гл. 2. При работе с этими и другими полноэкранными командами для ускорения процесса ввода данных в поля файлов БД с клавиатуры используется команда **SET CONFIRM ON [OFF]**, которая устанавливает режим автоматического перехода курсора в следующее поле после заполнения предыдущего (**OFF**) или оставляет курсор в последней позиции поля после его заполнения (**ON**). Команда **SET BELL ON** включает (устанавливается по умолчанию) предупреждающий звуковой сигнал при заполнении поля, вводе неверного значения поля и других ошибочных ситуациях (**SET BELL OFF** - отключает такое состояние). Для установки параметров звукового сигнала используется команда

SET BELL TO [<частота>,<длительность>]

Частота измеряется в герцах, а длительность в единицах, равных 0.0549 с. По умолчанию задана частота 512 Гц и длительность 2 единицы. Допустимый диапазон частоты от 19 до 10000 Гц, а длительности от 2 до 19 единиц. Значение звукового сигнала можно также установить с помощью параметра **BELL** в файле **Config.db**. Инициирование звукового сигнала выполняется функцией **CHR**(7) (см. разд. 3.7).

Примеры:

```
SET BELL TO && Восстанавливает значение сигнала по умолчанию
SET BELL TO 500,12
```

```
? CHR(7)    && Инициирует звуковой сигнал
```

Команды **BROWSE** и **EDIT** обеспечивают обновление содержимого полей в режиме ввода с клавиатуры. В ряде случаев возникает необходимость выполнять при обновлении многих записей стереотипные изменения, например умножить значение поля на одну величину и т.д. Для автоматизации такого вида обновления данных файла БД служит команда

UPDATE ON <ключевое поле> FROM <алиас>
REPLACE <имя поля 1> WITH <выражение 1>
[,<имя поля 2> WITH <выражение 2>...]
[RANDOM]

Эта команда заменяет содержимое полей исходного файла БД на данные, получаемые в результате вычисления (раскрытия) выражения, указанного в опции **WITH**. В это выражение могут входить поля исходного файла, а также и файла из другой рабочей области, указанной опцией **FROM**. Последний файл также служит для отбора обновляемых записей в исходном файле посредством сравнения значений их общего поля (с одинаковым именем), которое указывается опцией **<ключевое поле>**. При совпадении значений этого поля в текущих записях этих файлов выполняется замена содержимого полей текущей записи исходного файла значениями указанных выражений. Если значение ключевого поля не уникально, то операция замещения выполняется только для первой записи с одинаковым значением ключевого поля.

Оба файла должны быть либо отсортированы в восходящем порядке, либо проиндексированы по ключевому полю, если не задана опция **RANDOM**. В последнем случае только исходный файл должен быть проиндексирован по ключевому полю, а файл, указанный в опции **FROM**, может быть в любом порядке. Не рекомендуется обновлять ключевое поле, хотя это и не запрещается, поскольку по смыслу оно должно быть неизменным для обеспечения процесса поиска совпадающих полей.

Если выражение в опции **WITH** включает поле из файла, указанного в опции **FROM** (что часто используется при обновлении), то его имя нужно записывать в виде: **<алиас> -> <имя поля>**.

Пример:

```
USE Finans ORDER Nom
REPLACE ALL Total WITH 0
USE Fio ORDER Nom IN 2
UPDATE ON Nom FROM Fio REPLACE Total;
WITH Fio->Oklad*Fio->Proc
LIST
```

Если требуется обновить содержимое полей файла БД без привлечения данных из другого файла, например умножить содержимое числового поля на заданную величину, соединить содержимое двух символьных полей и т.д., можно использовать команду

**REPLACE [<границы>] <поле> WITH <выражение>
[, <поле> WITH <выражение...]
[WHILE <условие>] [FOR <условие>] [ADDITIVE]**

которая изменяет (замещает) значения заданных полей (включая поля **Memo**) текущей записи или группы записей, определенных границами и опциями **WHILE, FOR**. Поле, значение которого обновляется, и выражение, указанное в опции **WITH**, должны иметь один и тот же тип данных. Для числовых полей значение выражения, указанное в опции **WITH** может превышать длину поля. В этом случае значение поля записывается в формате с плавающей запятой. При записи содержимого полей **Memo** в обычные поля часть поля **Memo**, не укладывающаяся в обычное поле, обрезается. Опция **ADDITIVE** позволяет не замещать, а добавлять текст, заданный опцией **WITH** к содержимому полей **Memo**. Обновление значений проиндексированных полей файла БД вызывает автоматическое обновление файла индексов. Обновленная запись изменяет свое положение в индексном файле. Если в команде **REPLACE** заданы границы и опции **WHILE, FOR**, то могут быть обновлены не все записи. Например, если выполняется команда **REPLACE ALL**, то обновляется только первая запись и записи, значения которых следуют за новым значением этой записи. Это происходит вследствие того, что при обновлении выполняются изменения в индексе и указатель записи устанавливается на следующую за обновленной запись. Для гарантированного обновления значений всех записей следует использовать файл БД без индекса или установить на время обновления команду **SET ORDER TO 0**, а затем перестроить индекс командой **REINDEX**. Команда **REPLACE** не изменяет положение указателя записи, если не заданы границы обработки файла, опции **WHILE, FOR**.

Когда указатель записи стоит в конце файла (**EOF() = .T.**), то обновление не выполняется, если в команде не указаны границы обработки файла или опция **FOR**. Перед выполнением команды **REPLACE** для ее активизации следует установить указатель записи на начало файла с помощью команды **GO TOP**.

Примеры:

```
USE Finans
REPLACE Percent WITH 9.5 FOR Percent = 5.0
REPLACE ALL Unitcost WITH Unitcost * 1.1,;
Percent WITH Percent * 1.05
```

При использовании команд обработки файлов БД, которые включают списки полей (**BROWSE**, **EDIT**, **DISPLAY** и др.), можно предварительно выполнить специальную команду по установке тех полей, которые требуется вывести на экран. Эта команда имеет три варианта:

SET FIELDS ON (OFF)
SET FIELDS TO [<поле> [/R]/<имя вычисляемого поля>...]
[,<поле>[/R]/<имя вычисляемого поля>...]
SET FIELDS TO ALL [LIKE/EXCEPT <шаблон>]

Команда **SET FIELDS TO** определяет список доступных для вывода и обновления полей (включая вычисляемые поля) одного или нескольких файлов, а также позволяет устанавливать поля, которые соответствуют заданному шаблону. Кроме того, имеется опция **R** для установки на поле флажка "только чтение". Команда **SET FIELDS ON** активизирует установку полей.

Команды **SET FIELDS TO ALL** и **SET FIELDS OFF** делают доступными все поля активного файла БД. Опция **ALL LIKE** выводит поля, имена которых соответствуют заданному шаблону, а опция **ALL EXCEPT** - поля за исключением заданных в шаблоне. Команда **SET FIELDS TO** без опций отменяет сделанные ранее установки полей. После определения списка доступных полей последущие команды **SET FIELDS TO** добавляют в этот список имена новых полей. Если установлено значение **SET FIELDS ON**, то в командах **LIST STRUCTURE** и **DISPLAY** имена полей из заданного списка помечаются знаком >.

Команда **SET FIELDS TO** позволяет включить в список поля из нескольких файлов. Заметим, что связь между файлами при этом не устанавливается. Она устанавливается командой **SET RELATION**. Если среди файлов есть поля с одинаковыми именами, то нужно перед ними указывать алиасы рабочих областей со стрелкой вправо, например, **A-><имя поля>**. Список полей из команды **SET FIELDS TO** используется всеми командами, в которых можно задавать имена полей: **AVERAGE**, **BROWSE**, **CHANGE**, **COPY TO**, **COPY STRUCTURE**, **DISPLAY**, **EDIT**, **JOIN**, **LIST**, **SUM**, **TOTAL**.

Для обеспечения целостности данных в выборках и связанных файлах при установке команды **SET FIELDS** целесообразно добавлять записи по отдельности в каждый файл БД в состоянии, когда доступны все его поля. При изменении значений данных в индексированных полях связанных файлов БД следует помнить, что изменение данных в родительском файле без соответствующего изменения данных в сыновних файлах может привести к утрате связи между ними. Таким образом, в dBASE IV не обеспечивается автоматическое обновление данных в связанных файлах. Команда **SET FIELDS** не влияет на использование индексов, даже если индексные выражения содержат поля, не указанные в установленном списке. Этот список также не влияет

на команды **LOCATE, SET FILTER** и **SET RELATION**, которым доступны все поля в открытых файлах БД.

Для того чтобы связать файлы БД по общему ключевому выражению, используется команда

SET RELATION TO
SET RELATION TO <ключевое выражение> INTO <алиас>
[,<ключевое выражение> INTO <алиас>]...

Эта команда связывает активный файл БД с открытыми файлами в других рабочих областях, которые указываются алиасами в опции **INTO**. Активный файл БД, который управляет связью, называется родительским, а связываемые с ним файлы - сыновними. Команда **SET RELATION TO** без опций отменяет установку связи в текущей рабочей области. В качестве ключевого выражения можно использовать общее поле или комбинацию полей, например сцепление символьных полей или их подстрок, а также любые другие разрешенные выражения. Кроме этого типа связи в качестве ключевого выражения можно использовать функцию **RECNO()**, обеспечивающую связь по физическому номеру записи. Если при работе со связанными файлами БД не удается найти запись в сыновнем файле, то указатель записи в нем устанавливается в конец файла и значение функции **EOF()** равно **.T.** В сыновних файлах не действует команда **SET NEAR**.

Напомним, что связанные файлы образуют выборку. Для ее сохранения (вместе с установками среды) используется команда **CREATE VIEW <имя выборки> FROM ENVIRONMENT**. Эта команда создает и сохраняет на диске файл выборки (с расширением **.vue**) (совместимый с dBASE III+), в котором записывается следующая информация активной текущей среды:

все открытые файлы БД, индексные файлы и рабочие области каждого файла;

все связи между файлами БД;

номер текущей рабочей области;

список активных полей файлов;

открытый файл экранной формы (**.fmt**);

установленный фильтр.

Если установлена команда **SET CATALOG ON** и открыт каталог, то выборка автоматически записывается в него. После сохранения эту выборку можно активизировать командой **SET VIEW TO <имя выборки>**. Для деактивизации файла выборки можно открыть другой файл выборки или выполнить команду **CLOSE DATABASES**, которая закрывает все открытые файлы БД и связанные с ними файлы, образующие выборку. Формирование и сохранение выборок можно выполнить с помощью команды **CREATE/MODIFY VIEW**, меню которой описано в разд. 2.4.

Пример:

```
SELECT 1
USE Fio ORDER Fam
SELECT 2
USE Finans
SELECT 3
USE Kod ORDER Fam
SELECT 2
SET RELATION TO Fam INTO Fio, Fam INTO Kod
LIST Fio->Fam, Kod->Nom, Pay
```

Для обеспечения множественной обработки записей в связанных файлах БД используется команда **SET SKIP TO [<алиас> [,<алиас>]...]**. Она также позволяет осуществить доступ ко всем записям связанных файлов БД, которые удовлетворяют заданному значению ключевого выражения (по которому установлена связь командой **SET RELATION**). Обработка записей в связанных файлах БД выполняется в последовательности, определенной цепочкой алиасов, начиная с сыновних файлов и кончая родительским файлом. Родительский файл может быть первым в цепочке или же вообще не включаться в нее, но в любом случае он обрабатывается последним.

Пример:

```
SELECT 3
USE Kod ORDER Fam
SELECT 2
USE Finans ORDER Fam
SET RELATION TO Fam INTO Kod
SELECT 1
USE Fio
SET RELATION TO Fam INTO Finans
SET SKIP TO Kod,Fio,Finans
LIST Fio,Finans->Pay,Kod->Nom
```

При использовании команд **APPEND, BROWSE, EDIT** на экране (чаще всего в нижней строке экрана - строке сообщений) появляются различного рода сообщения и подсказки. В ряде случаев (например, когда пользователь уже хорошо освоил эти команды и хочет их модифицировать) возникает желание не выводить эти сообщения. Для этого используется команда **SET INSTRUCT ON (OFF)**, которая устанавливает режим их высвечивания (**ON**) или подавления (**OFF**).

После ввода данных в файл БД нередко требуется быстро их просмотреть. Для этого служат команды

LIST/DISPLAY [<границы>] [[FIELDS] <список выражений>] [OFF][<диапазон>] [FOR <условие>] [WHILE <условие>] [TO PRINTER/ TO FILE <имя файла>]

Эти команды выводят все поля файла БД, если их список не определен опцией **FIELDS**. В начале каждой записи, если не указана опция **OFF**, выводится ее номер. Как уже упоминалось, опции **<диапазон>**, **FOR** и **WHILE** определяют диапазон выводимых записей (см. раздел 3.1). Кроме них отбор выводимых записей определяется также предварительной установкой команд **SET FILTER** и **SET DELETED**. Первая команда подключает заранее разработанный файл запроса, а вторая устанавливает режим пропуска записей, помеченных для удаления. Команды **LIST** и **DISPLAY** идентичны с той разницей, что первая выводит записи непрерывно, а вторая делает остановку после вывода одной страницы экрана. Команда **SET PAUSE ON** (**OFF** - по умолчанию) также позволяет выводить любую информацию страницами экрана. При выводе данных файла БД на каждой экранной странице высвечиваются заголовки полей. Эта команда, кроме того, определяет постраничный вывод данных командой **SELECT** языка **SQL** (см. гл. 4).

Действие команды **SELECT** в языке **SQL** при **SET PAUSE ON** аналогично команде **DISPLAY**, а при **SET PAUSE OFF** - команде **LIST**.

Содержимое поля **Memo** выводится, если его имя указано в списке полей. Вывод выполняется в колонках шириной **50** позиций. Ширину вывода можно изменить командой **SET MEMOWIDTH TO <выр.N>**. Если имя поля **Memo** не указано в списке полей, то вместо содержимого поля выводится слово (маркер) **memo** или **MEMO**. Прописные буквы указывают на заполненность, а строчные на отсутствие содержимого поля **Memo**. Если записи имеют длину больше 80 символов, то они разворачиваются на экране с разрывом содержимого на 80-й позиции. Опции **TO PRINTER** и **TO FILE** позволяют вывести файл на принтер или записать на диск. Если перед выводом установлена команда **SET HEADING ON**, то высвечиваются имена полей. Эта команда выводит имена полей и при выполнении других команд, например **SUM, AVERAGE**. Для вычисляемых полей в качестве имен используется вычисляемое выражение. Если слово **FIELDS** записывается строчными (прописными) буквами, то и заголовки выводятся строчными (прописными) буквами.

Кроме вывода файлов БД команды **LIST** и **DISPLAY** позволяют также высвечивать данные о файлах в директориях, содержимое буфера **HISTORY** (хранилище выполненых с клавиатуры команд), данные об используемых переменных и массивах, информацию о текущем сеансе работы с dBASE IV, структуру указанного файла БД и имена рабочих станций, работающих в среде ЛВС. Для вывода информации о файлах в директориях аналогично команде **DOS dir** можно использовать команду

```
LIST/DISPLAY FILES [LIKE <шаблон>]
                          [TO PRINTER/TO FILE <имя файла>]
```

В этой команде шаблон предназначен для вывода файлов с именами (включая расширение), имеющими общую часть. Если не задано имя файла или его шаблон, то выводится информация о всех файлах БД в текущем директории, включая имя файла, число записей в нем, дату последнего обновления, размер файла в байтах, полное число высвеченных файлов, общий объем занимаемой памяти и объем свободной памяти на диске.

Следует отметить, что данные об открытых файлах могут не отражать текущее их состояние (число записей и объем), если установлена команда **SET AUTOSAVE OFF**. В этом случае актуальность данных восстанавливается после закрытия файлов.

Как уже упоминалось, все выполняемые в режиме набора с клавиатуры команды заносятся в специальный буфер, называемый **HISTORY**. Эти команды можно вызвать из буфера, что полезно при отладке программ, так как экономит время на набор команд. Для просмотра содержимого буфера **HISTORY** используется команда

LIST/DISPLAY HISTORY [LAST <выр. N>]
[TO PRINTER/TO FILE <имя файла>]

которая выводит список всех выполненных в режиме набора с клавиатуры команд (или последние команды, число которых задается опцией **LAST**), которые были записаны в этот буфер. По умолчанию в буфере помещается **20** команд. Его объем можно изменить командой **SET HISTORY**.

При разработке больших программ важно экономно расходовать оперативную память. Для этого используются различные способы оптимального распределения памяти. При этом нужно знать, сколько оперативной памяти занимают различные компоненты программы. Для оценки распределения оперативной памяти можно использовать команду

LIST/DISPLAY MEMORY [TO PRINTER/TO FILE <имя файла>]

которая выводит информацию об используемых активных переменных и элементах массивов: имя, статус (**PUBLIC** или **PRIVATE**), тип и значение переменной, а также имена, определения и емкость памяти, занимаемой активными окнами, вертикальными и горизонтальными меню и их позициями, значения всех системных переменных, а также емкость остающейся памяти. Максимальное количество переменных определяется установками параметров **MVBLKSIZE** и **MVMAXBLKS** в файле **Config.db**. Первый параметр определяет размер блока памяти переменных в единицах (участках памяти по **56** байт), отводимых для одной числовой, логической или типа даты переменной, а второй - максимальное число таких блоков. По умолчанию устанавливается **MVBLKSIZE = 50** и **MVMAXBLKS = 10**, что позволяет хранить до 500 числовых переменных. Для массивов переменных один такой участок отводится под его имя, а специальный блок

участков - под элементы. Символьные переменные и массивы занимают несколько участков в зависимости от длины содержимого. В целях обеспечения поиска ошибок при выполнении команд для каждой переменной и поля файла БД отводится дополнительный участок памяти для отладочной информации. Общая емкость этой памяти определяется параматрами **RTBLKSIZE** и **RTMAXBLKS**, задаваемыми в файле **Config.db**. Первый параметр определяет размер блока памяти для отладочной информации в единицах по **17** байт, а второй - максимальное число таких блоков. По умолчанию **RTBLKSIZE = 50** и **RTMAXBLKS = 10**. Оперативная память занимается блоками переменных и отладочной информации по мере их объявления и заполнения очередного блока. Кроме участков памяти, отводимых под содержимое переменных и отладочную информацию (кроме символьных переменных), имеются участки памяти, обеспечивающие их вывод. Для каждой переменной типа даты требуется еще **8** байт, для логической переменной **1** байт и для числовой - **20** байт. В последней строке выведенной командой **LIST/DISPLAY MEMORY** информации содержится общая емкость дополнительной памяти, требуемой для высвечивания переменных. В эту емкость также входит емкость памяти, занимаемая системными переменными, равная **264** байтам. Такое подробное описание памяти, выделяемой для переменных, полезно для определения ее оптимальной емкости и распределения между переменными и другими компонентами программы, например окнами или меню.

Полную информацию о состоянии среды можно получить командой **LIST/DISPLAY STATUS**, которая выводит: номер текущей рабочей области, имя файла БД, его алиас и маршрут по директориям, признак установки режима только чтения файла (если установлен), открытые индексные файлы (типа **.ndx** и **.mdx**), индексные выражения для каждого индексного файла и указателя, их опции **UNIQUE** и **DESCENDING**, открытые файлы полей **Memo**, формулы фильтров для отбора записей, связи файлов БД, файлы экранных форм, маршруты для поиска файлов, диск, установленный по умолчанию, устройство для вывода печатаемых материалов, загруженные программные модули, текущую выбранную рабочую область, левую границу вывода на печать, текущий открытый файл процедур, счетчик повторения захватов, счетчик количества проверок обновления файлов, значение параметра **DEVICE** (**SCREEN**, **PRINT** или **FILE**), символ валюты, символ разделителя полей файла БД, количество открытых файлов, установки команды **ON**, текущие значения **ON** или **OFF** большинства **SET**-команд и назначения функциональных клавиш.

В сложных информационных системах количество файлов БД бывает достаточно велико. Нередко возникает необходимость

просмотреть структуру файла и вывести ее на печать. Для этого служит команда

LIST/DISPLAY STRUCTURE [IN <алиас>] [TO PRINTER/ TO FILE <имя файла>]

которая выводит информацию о файле БД в текущей рабочей области или заданной алиасом: имя файла, число записей, дату последнего обновления, полное определение каждого поля (имя, тип, ширину и т.д.), поля, являющиеся указателями связанного файла **.mdx** и общий размер записи в байтах, который складывается из размеров полей плюс один байт, служащий для пометки записи на удаление. Опции **TO PRINTER** и **TO FILE** для всех команд **LIST/DISPLAY** позволяют вывести информацию на принтер или записать в файл на диск.

Команды **BROWSE** и **EDIT** позволяют выполнять все действия, необходимые для ввода, обновления, удаления и добавления записей с использованием меню этих команд, которые дают возможность передвигатся по файлу, помечать записи для их последующего удаления, вводить и обновлять с клавиатуры данные. Кроме этих команд существуют специальные команды перемещения по файлу, добавления новых записей и пометки записей для их последующего удаления. Обычно эти команды используются в программах для обеспечения автоматизированной обработки информации, но могут также применяться в режиме набора с клавиатуры. Перед тем как выполнять команды в этом режиме, необходимо файл БД открыть. После этого можно выполнять с ним операции. Для перемещения по файлу используется команда

GO/GOTO BOTTOM/TOP [IN <алиас>]

или

GO/GOTO [RECORD] <номер записи> [IN <алиас>]

или

<номер записи>

Эта команда осуществляет переход на указанную запись в активном файле БД. Если индексный файл открыт, опции **TOP** и **BOTTOM** осуществляют переход к первой и последней записям индексного файла. Если же индексный файл не используется, эти опции осуществляют переход к первой и последней физическим записям файла БД. Для выполнения перехода к записи по физическому номеру используется опция **GO** или просто ее параметр **<номер записи>**.

В программах часто требуется перемещать указатель записи не на конкретную запись, а на следующую после текущей. Такое перемещение необходимо при организации циклов обработки файла БД. Эту операцию выполняет команда **SKIP [<выр.N>] [IN <алиас>]**. Указатель записи в активном файле БД в текущей рабочей области (или в области заданной опцией **IN**) можно

перемещать вперед или назад в зависимости от знака числового выражения, определяющего количество пропускаемых записей. Если с файлом БД используется индекс, указатель перемещается в соответствии с индексом. Для неиндексированных файлов БД указатель перемещается по физическим номерам записей. Если выражение не указано, указатель перемещается вперед на одну запись. Опция **IN** позволяет перемещать указатель записи в другой рабочей области без ее выбора командой **SELECT** в качестве текущей. При этом указатель записи в текущей рабочей области не изменяется. Если установлена команда **SET TALK ON**, то при высвечивании данных на экране выводится номер записи.

При работе с индексными файлами в среде ЛВС следует быть особенно внимательными, выполняя команду **SKIP** (более подробно см. разд. 3.7). Если один из пользователей изменяет значение индексированного поля файла БД или добавляет запись к нему, то после этого весь файл переупорядочивается в соответствии с установленным главным индексом. Это надо иметь в виду при установке командой **SKIP** указателя на обновленную запись или запись, следующую после нее, поскольку ее расположение уже изменено в результате появления обновленной или новой записи. Напомним, что упорядочивание индексированного файла БД означает не физическое изменение положения записей, а соответствующее построение индексного файла, который определяет вывод данных. Физическое расположение записей при этом не меняется.

Например, если один пользователь добавляет запись в файл, а другой пытается командой **SKIP** перейти на эту запись, то указатель записи будет показывать конец файла. Если же один пользователь обновляет значение индексированного поля некоторой записи, а другой пытается перейти на следующую за ней запись, то указатель записи также будет установлен не на эту, а на совершенно другую запись. Для того чтобы избежать таких неприятностей, можно сначала выполнить команду **GO RECNO()** для установки указателя на текущую запись, а затем - команду **SKIP**.

При использовании команды **SKIP** в программах для перемещения по файлу часто требуется получить признак его начала или конца. Для этого применяются функции **BOF()** и **EOF()**. Функция **BOF([<алиас>])** определяет начало файла и возвращает логическое значение .T., если предпринимается попытка переместить указатель перед первой записью, в противном случае возвращается .F. Следует отметить, что значение .F. возвращается и в том случае, если вообще нет открытого файла БД. Аналогично индицирует конец файла функция **EOF([<алиас>])**. В программах и режиме выполнения команд с клавиатуры нередко требуется определить текущий номер записи. Это осуществляется функцией **RECNO([<алиас>])**, которая возвращает текущий номер в указанном файле БД. Если

алиас не задан, то подразумевается текущая рабочая область. Возвращается 1, если в файле БД записей нет, и 0, если нет открытых файлов БД. Если указатель записи перемещается на следующую после последней запись в файле БД, то возвращается число, на единицу большее количества записей в файле и **EOF() = .T.** Если указатель записи перемещается на запись, предшествующую первой записи файла БД, что эквивалентно выполнению команды **SKIP-1**, то возвращается **1** и **BOF() = .T.**

Примеры:

```
USE Fio
SKIP                 && Пропуск одной записи вперед
STORE 5 TO mskip     && Для пропуска пяти записей
SKIP mskip           && вперед используется переменная
SKIP-3               && Переход на три записи назад
USE Finans IN 2      && Перемещает указатель записи в
SKIP 4 IN Finans     && файле БД Finans, тогда как
*                       активным является файл БД Fio
```

После того как указатель записи перемещен на нужную запись, которая становится текущей, можно выполнить требуемую операцию, например ввести запись до или после текущей. Для этого используется команда **INSERT [BEFORE] [BLANK]**, которая позволяет ввести новую запись (или пустую запись, если указана опция **BLANK**) после текущей или перед ней (если задана опция **BEFORE**). Пометить одну или несколько записей для их последующего удаления можно командой **DELETE [<диапазон>] [FOR<условие>] [WHILE<условие>]**. Опции **<диапазон>**, **FOR** и **WHILE** позволяют помечать записи, отобранные в соответствии с заданным диапазоном и условиями.

В программах нередко требуется определить, помечена ли текущая запись для удаления. Это можно сделать с помощью функции **DELETED([<алиас>])**, которая возвращает логическое значение **.T.** для помеченной записи.

Для того чтобы уничтожить пометки на удаление записей используется команда

RECALL [<диапазон>] [FOR<условие>] [WHILE<условие>]

Опции **<диапазон>**, **FOR** и **WHILE** как обычно определяют дипазон, условия отбора записей и прекращения этой операции. Перед уничтожением пометок на удаление должна быть установлена команда **SET DELETED ON**. Эта команда включает (**OFF** выключает) режим пропуска помеченных на удаление записей при обработке файла БД многими командами dBASE IV (например, **LOCATE**, **LIST**, **RECALL** и др.) кроме команд **INDEX**, **REINDEX**, **DISPLAY** и **GOTO**, которые обрабатывают, выводят и устанавливают указатель на любую запись независимо от установки **SET DELETED**. Если установлена команда **SET DELETED**

ON, то пометку одной записи с номером **N** можно уничтожить командой **RECALL RECORD N**.

Примеры:

```
USE Fio
RECALL              && Уничтожает метку удаления текущей записи
RECALL RECORD  10   && Уничтожает  метку удаления записи 10
RECALL ALL          && Уничтожает все метки удаления записей
*                      активного файла БД
```

Физическое удаление помеченных записей активного файла БД выполняет команда **PACK**. При этом происходит автоматическое перестроение всех открытых индексов (выполняется команда **REINDEX**). После выполнения команды **PACK** и закрытия файла, дисковое пространство, занятое исключенными записями, становится доступным для ввода. Для удаления всех записей активного файла БД без их предварительной пометки используется команда **ZAP**. Удаление всех записей можно также осуществить с помощью последовательного выполнения команд **DELETE ALL** и **PACK**, но команда **ZAP** работает быстрее. Следует отметить, что индексные файлы, связанные с уничтоженным файлом БД, остаются и переиндексируются на оставшуюся одну пустую запись файла БД.

Важной операцией при работе с файлами БД является добавление записей в конец. Она обычно используется при первоначальном вводе данных, а также и в процессе эксплуатации информационной задачи. Как уже отмечалось, команда **BROWSE** позволяет выполнять все основные операции по вводу, обновлению и добавлению данных. Кроме этой команды в языке dBASE IV имеются специальные команды для добавления записей в конец файла БД с клавиатуры, из другого файла, из массива данных и т.д., которые могут использоваться в программах и в командном режиме. Команда **APPEND [BLANK]** позволяет добавлять записи, введенные с клавиатуры в конец активного файла БД (или пустую запись, если указана опция **BLANK**). По умолчанию для ввода используется экранная форма команды **EDIT** или экранная форма, установленная командой **SET FORMAT TO**. При добавлении записей все активные индексные файлы автоматически обновляются. Если установлено состояние **SET AUTOSAVE ON**, то запись на диск осуществляется автоматически после каждой единичной операции ввода-вывода, кроме последней записи, которую нужно сохранить клавишами **Ctrl-End**. В состоянии **SET AUTOSAVE OFF** (устанавливается по умолчанию) добавленные записи сохраняются автоматически только после заполнения буфера. Состояние **ON** или **OFF** этой команды может быть установлено в командном режиме с помощью команды **SET** и в файле **Config.db**. При выполнении операций по обновлению данных нередко требуется перенести значения предыдущих

полей записи в последующие. Для этого можно использовать команду

SET CARRY TO [<список полей> [ADDITIVE]]

которая копирует данные из предыдущей записи в последующую запись при обновлении данных командами **BROWSE, APPEND** или **INSERT** (кроме команд **INSERT BLANK** и **APPEND BLANK**). При задании списка полей копируются только указанные в нем поля. В этом случае команда **SET FIELDS** не действует. Опция **ADDITIVE** добавляет значения полей из предыдущей записи к соответствующим полям последующей записи. Если список полей не указан, то копируются все поля. Состояние **SET CARRY ON** (**OFF** устанавливается по умолчанию) включает (выключает) копирование полей.

Для добавления записей в активный файл БД из другого файла, в том числе не относящегося к dBASE, используется команда

APPEND FROM <имя файла>/?
[[TYPE] <тип файла>] [FOR <условие>]

Опция **<тип файла>** имеет следующие значения:

dBASEII - файл dBASE II;

DELIMITED - файл ASCII формата DELIMITED;

DELIMITED WITH BLANK - файл с полями, разделенными одним пробелом, и записями, разделенными знаком возврата каретки и конца строки;

DELIMITED WITH <разделитель> - файл с полями, отделенными запятыми и символьными данными, заключенными в знак разделителя (по умолчанию кавычки);

DIF - файл формата VisiCalc;

FW2 - файл БД или электронной таблицы Framework II;

RPD - файл данных RapidFile;

SDF - файл ASCII системного формата;

SYLK - файл электронной таблицы MultiPlan;

WKS - файл электронной таблицы Lotus 1-2-3, версия 1A.

Строки данных в электронных таблицах интерпретируются как записи, а колонки в них, как поля.

В разд. 3.1 была отмечена функция переменных и массивов как временных промежуточных хранилищ данных. Данные из переменных и массивов можно передавать в файл БД и обратно. Для добавления данных из массива в конец активного файла БД используется команда

APPEND FROM ARRAY <имя массива> [FOR <условие>]

Строки массива переводятся в записи, а колонки в поля. Если в массиве больше колонок, чем полей в файле БД, то лишние колонки игнорируются. Если в файле БД больше полей, чем колонок в массиве, то лишние поля остаются пустыми. Каждый

элемент массива должен быть того же типа, что и тип поля, в которое он помещается. Если используется опция **FOR**, то условие оценивается перед помещением строки массива в файл БД. Условие должно включать имя поля из файла БД. Строка добавляется в том случае, если условие оценивается как **.T.** Если условие оценивается как **.F.**, то строка пропускается и обрабатывается следующая строка. Массив должен быть заранее объявлен и инициализирован.

Пример:

```
DECLARE Massiv[1,5]
Massiv[1,1]="Петров"
Massiv[1,2]="923-15-48"
Massiv[1,3]=DATE()
Massiv[1,4]=.T.
Massiv[1,5]=250.00
USE Kadr
APPEND FROM ARRAY Massiv
```

Предыдущие команды добавления записей в конец файла БД позволяют вводить данные и в поле **Memo**. Но из-за специфики данного поля используются специальные команды по его обработке, например, команда

APPEND MEMO <имя memo поля>
FROM <имя файла> [OVERWRITE]

Эта команда позволяет ввести данные, если поле пустое, добавить к имеющемуся содержимому текущего поля **Memo** данные или перекрыть содержимое, если используется опция **OVERWRITE**, данными из заданного текстового файла. Если не указывается расширение файла, то подразумевается **.txt**.

Пример:

```
USE Fio
APPEND MEMO person_at FROM Txtfile
```

К числу команд по созданию файлов БД можно также отнести команду **IMPORT FROM <имя файла> [TYPE] PFS/DBASE II/FW2/RPD/WK1** которая формирует файл БД dBASE IV с расширением **.dbf** (и присваивает ему имя исходного файла) из файлов формата PFS, DBASE II, FW2, RPD и WK1. Для импорта файлов dBASE II нужно изменить их расширение с **.dbf** на **.db2**.

Следует учитывать, что при импорте преобразование внешних файлов осуществляется в соответствии со структурой, размерами и типами файлов dBASE IV. При импорте файлов формата PFS создаются три типа файлов dBASE IV: файл БД (**.dbf**), исходный (**.fmt**) и компилированный (**.fmo**) файл экранной формы.

Выше рассматривалось упорядочивание вывода данных из файлов в соответствии с заданными индексами. При таком упорядочивании физическое расположение записей исходного

файла не меняется. В ряде случаев требуется создать физически упорядоченный файл. Для этого используется команда

SORT TO <имя файла> ON <поле1> [/A] [/C] [/D] [,<поле2> [/A] [/C] [/D]...][ASCENDING]/[DESCENDING] [<границы>] [WHILE <условие>] [FOR <условие>]

которая создает новый файл БД с записями, отсортированными в алфавитном, хронологическом или числовом порядке по заданным полям. Сортировка выполняется в порядке возрастания (**/A**) или убывания (**/D**) значений полей. Допускается сортировка максимум по десяти полям. Опция **/C** определяет сортировку независимо от прописных и строчных букв.

Запрещается сортировать по логическим полям и полям **Memo**, по подстрокам и выражениям. Для того чтобы получить отсортированный файл по подстрокам и выражениям, можно использовать две другие команды. Сначала команду **INDEX**, позволяющую создать индексный файл по заданному выражению, а затем команду **COPY**, которая создает копию файла с записями, упорядоченными по индексному выражению. Отсортированный файл БД нельзя записывать в себя или в любой другой открытый файл.

Ключевые слова **ASCENDING** и **DESCENDING** можно использовать в качестве альтернативы значениям **/A** и **/D**. Можно комбинировать опцию **/C** с опциями **/A** или **/D**. Если используется комбинация опций, нужно указывать только один разделитель (например, **/DC**). При сортировке по нескольким полям, наиболее приоритетное поле следует указывать первым.

Пример:

```
USE Fio
SORT ON Otd TO Otd1
```

После создания файла БД часто возникает необходимость его копирования, например для создания дубликата, с целью последующей модификации файла и т.д.

Команда **COPY FILE <имя файла 1> TO <имя файла 2>** делает копию файла любого типа. Имена копируемого (исходного) файла и выходного файла (куда копируется исходный файл) должны указываться с расширением, и при необходимости именам файла должен предшествовать идентификатор диска и маршрут по директориям (если используется диск и директорий, отличный от текущего). Не требуется указывать идентификатор диска, где размещен файл 1, если он задан с помощью команды **SET PATH** (или **PATH = <маршрут по директориям>** в файле **Config.db**). Запрещается копировать открытые файлы. При копировании файла БД, содержащего поля **Memo**, соответствующий файл **.dbt** должен копироваться отдельно. Это же относится и к связанному индексному файлу типа **.mdx**. Перед копированием целесообразно убедиться в том, что

на диске достаточно места для копии. Для этого можно использовать функцию **DISKSPACE()**, которая возвращает число свободных байтов на активном диске.

После копирования или в других случаях нередко требуется переименовать файл. Для этого используется команда

RENAME <имя файла> TO <новое имя файла>

в которой и старое, и новое имя файла должно включать расширение. Если файлы расположены на диске, отличном от текущего, то перед именем файла, как это уже неоднократно упоминалось, следует поместить идентификатор диска и маршрут по директориям. Новое имя файла не должно совпадать с именем существующего файла, расположенного в том же директории. При переименовании файла БД с полями **Memo**, файл **.dbt** следует переименовывать отдельно. Запрещается изменять имя открытого файла.

Пример:

```
RENAME Fio.DBF TO Spisok.DBF
```

Перед копированием иногда целесообразно определить объем файла БД. Его можно вычислить (без полей **Memo**) как произведение длины записи на их количество. Для определения длины записи можно использовать функцию **RECSIZE([<алиас>])**, а для определения числа записей - функцию **RECCOUNT([<алиас>])**. Функция **RECCOUNT** возвращает число записей в текущем, или указанном алиасом файле БД. Если нет открытых файлов, то возвращается 0. Возвращенное число включает все записи файла БД, в том числе и помеченные для удаления, даже если установлены команды **SET DELETED ON** и **SET FILTER**.

Пример:

```
COPY FILE PROG.PRG TO COPYPROG.PRG
```

При выполнении операций импорта, экспорта, перезаписи и удаления файла надо быть внимательными. Особенно это касается удаления, так как после этой операции файл восстановить уже нельзя. Для предотвращения от несанкционированной перезаписи или удаления файла служит состояние **SET SAFETY ON** (устанавливается по умолчанию), которое высвечивает при создании нового файла с уже существующим именем сообщение **File alredy exists** (файл уже существует) и предлагает на выбор меню с позициями: **Overwrite** (перезаписать) или **Cancel** (отменить). Если нельзя перезаписать этот файл, то после выбора позиции **Cancel** можно ввести имя нового файла, создав тем самым копию файла под другим именем. В состоянии **SET SAFETY OFF** перезапись осуществляется автоматически без предупреждающего сообщения. При выполнении команды **ZAP**

в состоянии **SET SAFETY ON** также выводится сообщение, требующее подтверждения об удалении файла.

Кроме копирования, новый файл можно создать слиянием полей из двух файлов. Эта операция выполняется командой

JOIN WITH <алиас> TO <имя файла>
FOR <условие> [FIELDS <список полей>]

которая создает новый файл БД посредством слияния заданных полей (кроме **Memo** полей) двух открытых файлов БД: одного в текущей рабочей области, а второго в рабочей области, заданной алиасом (напомним, что по умолчанию алиас совпадает с именем файла БД).

Опция **FOR** позволяет отобрать в новый файл записи из двух исходных файлов в соответствии с заданным условием, которое проверяется для каждой записи этих файлов следующим образом. Сначала проверяется условие для первой записи первого файла и для всех записей второго файла, и все удовлетворяющие этому условию пары записей (с заданными полями) включаются в новый файл. Затем этот процесс повторяется для второй записи и т.д.

Заметим, что для больших файлов процесс их слияния может занять довольно много времени, а новый файл потребует много места на диске, если условие отбора достаточно широкое. Если в исходных файлах есть поля с одинаковыми именами, которые нужно включить в новый файл, то надо в списке перед ними указать алиас со стрелкой вправо (для неактивной рабочей области). Вместо указания списка полей можно предварительно установить команду **SET FIELDS**.

Пример:

```
USE Person ALIAS B
USE FIO
JOIN WITH B TO SPISOK FOR Fam=B->Fam
SET FIELDS Fam, Name, B->Adres, B->Gorod, Summa
USE SPISOK
LIST STRUCTURE
```

Для того чтобы скопировать из файла БД, не все записи, а только часть и, кроме того, не все поля, а только заданные, используется команда

COPY TO <имя файла>
[TYPE <тип файла>] [<диапазон>]
[FIELDS <список полей>] [WHILE <условие>]
[FOR <условие>]

которая копирует в новый файл записи, определяемые заданным диапазоном, условиями **FOR** и **WHILE** и метками для удаления (если установлена команда **SET DELETED ON**). Копируемые поля определяются опцией **FIELDS** или командой **SET FIELDS TO**.

Выходной файл может быть типа dBASE и не-dBASE. В последнем случае требуется в опции **TYPE** задать тип внешнего файла. Поля **Memo** копируются только в том случае, если выходной файл является файлом dBASE IV. При копировании во внешние файлы не проверяется совместимость типов и форматов данных. В качестве типов внешних файлов можно использовать те же типы, что и в команде **APPEND FROM**. Файлы БД dBASE IV с полями **Memo** нельзя открыть в dBASE III+. Для этого их надо скопировать в новый файл, используя тип **DBMEMO3**. Копирование файлов DBASE IV в файлы формата **PFS** можно выполнить командой

EXPORT TO <имя файла> [TYPE] PFS/DBASE II/FW2/RPD
[FIELD <список полей>] [<границы>]
[FOR <условие>] [WHILE <условие>]

которая позволяет копировать файл БД, кроме того, и в файлы форматов **DBASE II**, **FW2** и **RPD**. Использование остальных опций описано в предыдущей команде **COPY**.

Чтобы скопировать содержимое только поля **Memo** в другой файл, используется команда

COPY MEMO <имя memo поля> TO <имя файла>
[ADDITIVE]

которая копирует информацию из текущего поля **Memo** в текущую запись файла на диске (если не указано расширение, то подразумевается текстовый файл). Для копирования поля **Memo** из другой рабочей области перед его именем следует указать алиас рабочей области со стрелкой. Если имя поля содержится в списке, определенном командой **SET FIELDS**, то алиас указывать не требуется. Опция **ADDITIVE** позволяет добавить содержимое поля **Memo** в конец записи или файла (если выходным файлом служит текстовый файл). Если эта опция не используется, а выходной файл существует на диске, то содержимое поля **Memo** заменяет (перекрывает) данные в этом файле.

Пример:

```
COPY MEMO Avtobio TO Avtofile ADDITIVE
```

Операцию, обратную команде **APPEND FROM ARRAY**, выполняет команда

COPY TO ARRAY <имя массива> [FIELDS <список полей>]
[<границы>] [FOR <условие>]
[WHILE <условие>]

которая копирует отобранные записи и содержимое заданных полей из файла БД в существующий массив. Поля записей копируются последовательно в колонки массива. Каждая запись становится строкой массива. Процесс копирования полей продол-

жается до тех пор, пока не исчерпаются поля или колонки массива. Если файл БД имеет больше полей, чем колонок в массиве, лишние поля не копируются. В противном случае лишние колонки массива остаются неизменными. Нельзя копировать в массив поля **Memo**. В одномерный массив копируется одна запись.

Пример:

```
USE Fio                              && Файл Fio.DBF имеет 5 полей
COUNT FOR Fio_id = 'A0001' TO mcnt
DECLARE Records [mcnt,5]             && Определение массива
COPY TO ARRAY Records FOR Fio_id = 'A0001'
```

Во многих операциях по обработке файла встречалась опция **FOR** для отбора записей по заданному условию. Если одно такое условие требуется использовать во многих командах, то вместо опции **FOR** можно предварительно выполнить команду

SET FILTER TO [FILE <имя файла>/?] [<условие>]

которая позволяет установить условие (фильтр) для отбора записей при выполнении многих команд обработки файла БД. Условие в этой команде представляет собой допустимое в dBASE IV выражение, которое может состоять из элементарных условий (см. разд. 2.4), объединенных логическими операторами.

Опция **FILE** дает возможность устанавливать в качестве фильтра файл запроса, созданный командой **CREATE/MODIFY QUERY** (см. раздел 3.3) или в режиме УЦ, управляемом меню. Если открыт каталог, то опция ? позволяет высветить список файлов запросов для активного файла БД. Файлы запросов, созданные в dBASE III+, можно использовать в dBASE IV, но их нельзя модифицировать.

Для каждой рабочей области можно установить свой фильтр. Выключение фильтра выполняется командой **SET FILTER TO** без опций. Фильтр активизируется только после перемещения указателя записи файла БД, например командами **GO TOP** или **SKIP**. Фильтр применяется не только к командам, которые имеют опцию **FOR**, но и ко всем командам, в которых используются открытые файлы БД (например, команды подготовки к печати данных из файла БД, команды вычислений агрегированных значений полей и т.д.).

Пример:

```
USE Fio
SET FILTER TO Otd = '01'
GO TOP
LIST Fam,Name
SET FILTER TO
```

После завершения операций с файлами БД их следует закрыть командой **CLOSE ALL/ALTERNATE/DATABASES/FORMAT/**

INDEX/PROCEDURE, которая позволяет закрыть все открытые файлы или файлы определенного типа.

Опция **ALL** закрывает все файлы и устанавливает в качестве текущей рабочую область 1. При этом если открыт каталог (для него предусмотрена рабочая область 10), то его рабочая область не закрывается. Для закрытия файлов определенного типа используются опции: **ALTERNATE** - альтернативные файлы, записанные на диск; **DATABASES** - файлы БД, индексные файлы, файлы полей **Memo** и экранных форм, связанные с файлами БД; **FORMAT** - файлы экранных форм; **INDEX** - индексные файлы; **PROCEDURE** - программные файлы и процедуры. Для закрытия конкретного файла БД и связанных с ним файлов используется команда **USE**.

Выше рассматривалась команда **ZAP** удаления записей в файле БД. После такого удаления структура файла сохраняется и можно вновь вводить данные. Если требуется полностью уничтожить файл, то это можно сделать командой

ERASE <имя файла>/? или DELETE FILE <имя файла>/?

При этом удаляемый файл должен быть закрыт, а имя файла должно включать расширение. Если файл удаляется не на текущем диске или в другом директории, то ему должны предшествовать имя диска и маршрут по директориям. При удалении файла БД с полями **Memo** файл этих полей следует удалять отдельно. Это же относится к связанному множественному индексному файлу. В отличие от аналогичной команды DOS в данной команде не разрешается использовать шаблон имени файла.

Перед выполнением любой работы на ПК целесообразно установить диск, который будет использоваться по умолчанию для всех операций (записи новых, модифицированных файлов БД, связанных и вспомогательных файлов и т.д.). Для этого служит команда **SET DEFAULT TO <диск>[:]**. При этом устанавливается директорий, назначение которого было последним для этого диска. Например, если был осуществлен переход на диск **A** с директория **\BASA** диска **C**, то после возвращения обратно на диск **C** командой **SET DEFAULT** будет установлен директорий **\BASA**. Команда **SET DEFAULT** не проверяет, существует ли назначаемый диск, и не меняет диск, на который осуществляется переход после выхода из dBASE IV.

При разработке программ для работы с файлами весьма полезной является также команда **SET PATH TO [<список маршрутов>]**, которая позволяет установить список маршрутов по директориям дисков, используемых для поиска требуемого файла, если он не найден в текущем директории.

Следует отметить, что в dBASE IV не используются маршруты, установленные в DOS командой **path**, так же как и в DOS не

используются маршруты, установленные в dBASE IV. Точно так же маршруты, установленные в УЦ, не используются в командном режиме и при выполнении программ, как и маршруты, установленные командой **SET PATH** в программах не используются в УЦ. Команда **SET PATH TO** без параметров восстанавливает директорий, установленный по умолчанию. Этим директорием является директорий, откуда стартовал dBASE IV, директорий, установленный командой DOS **chdir** или командой dBASE IV **SET DEFAULT**.

Для вывода имен файлов в директориях и информации о файлах на диске используется команда

DIR/DIRECTORY [[ON] <диск:>] [[LIKE]
[<маршрут>][<шаблон>]

Если не задано имя файла или его шаблон, то выводится информация только о файлах БД. Если не указан диск и маршрут, то выводится информация в текущем директории на текущем диске. Для файлов БД выводятся их имена, число записей, дата последнего обновления, размер в байтах, общее число высвеченных имен файлов и емкость оставшейся свободной памяти на диске. Для других типов файлов высвечивается только имя.

Следует отметить, что директорий может не отражать последних изменений, если установлена команда **SET AUTOSAVE ON**, до тех пор, пока файл не будет закрыт. Для вывода имен всех типов файлов следует использовать шаблон ***.***.

3.3. ПОИСК И ОТБОР ДАННЫХ

Описанные в разд. 3.2 команды позволяют выполнить все операции по созданию файлов БД, вводу в них данных, последующего обновления и удаления записей, а также просмотра записей. Для передвижения по файлу предусмотрены позиции меню полноэкранных команд типа **BROWSE** и **EDIT**, а также команды пошагового перемещения и перемещения на запись с заданным номером. Однако нередко требуется найти запись не по ее положению в файле БД, а по содержанию полей. Такой вид поиска предусмотрен, например, в меню команды **BROWSE**.

В настоящем разделе будут рассмотрены специальные команды для организации поиска и отбора записей из файлов БД. Способ поиска по содержанию полей зависит от того, созданы ли индексные файлы по этим полям или их комбинациям. Для поиска по неиндексированным полям используется команда

LOCATE [FOR] <условие>[<диапазон>]
[WHILE <условие>]

Эта команда выполняет в активном файле БД поиск первой записи, удовлетворяющей условию, заданному опцией **FOR**. Опции **<диапазон>** и **WHILE** определяют диапазон и условие начала и конца просмотра записей. Если эти опции не заданы, то поиск начинается с первой записи. При успешном поиске указатель записи устанавливается на запись, удовлетворяющую заданному условию. Для определения успешности поиска можно использовать функцию **FOUND()**, которая возвращает в этом случае логическое значение **.T.** В случае установки команды **SET TALK ON** высвечивается номер найденной записи. Если в заданном диапазоне или в целом файле требуемая запись не найдена, то высвечивается сообщение **End of LOCATE scope** (конец поиска в заданном диапазоне) и указатель записи перемещается в конец файла (за последнюю запись). В этом случае функция **EOF()** принимает значение **.T.**, а **FOUND() = .F.** Для поиска следующей записи, удовлетворяющей заданному условию, нужно вслед за командой **LOCATE** выполнить команду **CONTINUE**. В любой из рабочих областей можно организовать автономный поиск. Для этого выбирается другая рабочая область и выполняется команда **LOCATE** (и, если нужно, **CONTINUE**). После возвращения в исходную рабочую область указатель записи в ней находится в том же положении, какое было до перехода, и можно продолжать прерванный поиск.

Для поиска по индексированным полям используются команды **SEEK** и **FIND**. Команда **SEEK <выражение>** выполняет поиск первой записи файла БД с заданным значением индексированного поля. При этом соответствующий индекс должен быть установлен в качестве главного (управляющего) индекса.

Заданное поисковое значение может быть результатом раскрытия любого, разрешенного в dBASE IV выражения. Обычно это выражение записывается в переменную. Типы данных поискового значения и индексного выражения должны совпадать. В качестве поискового значения можно задавать подстроку (часть строки) символьного поля, но начиная только с первой позиции. Причем в этом случае должно быть установлено состояние **SET EXACT OFF** (задается по умолчанию). Это состояние обеспечивает сравнение строк разной длины. Сравнение двух строк выполняется побуквенно, начиная с первой позиции и до конца правой из двух сравниваемых строк. Если эти символы совпадают, то результат сравнения положительный (возвращается логическое значение **.T.** этой операции). В данном случае правой является строка заданного значения в команде **SEEK**, а левой - значение из соответствующего поля файла БД. Например, если требуется найти значение "Петров" поля **FAM** (фамилия) файла БД, то для поиска можно использовать часть этого значения, начиная с первой позиции - "Пет", если, конечно, это сокращение однозначно идентифицирует данную фамилию.

В состоянии **SET EXACT ON** процесс сравнения строк выполняется аналогично, но до конца самой длинной из строк. В состоянии **SET DELETED ON** и при установленной команде **SET FILTER** для поиска отбираются отфильтрованные и непомеченные для удаления записи. Если запись не найдена, то высвечивается сообщение **Find not successful** (поиск неуспешен) и указатель записи перемещается в конец файла БД (**EOF() = .T.**), если установлена команда **SET NEAR OFF**. При успешном выполнении команды **SEEK** функция **FOUND()** принимает значение **.T.** и указатель записи остается на найденной записи. Для подавления вывода сообщения можно использовать установку состояния **SET TALK OFF**. Состояние **SET TALK ON** позволяет выводить на экран информацию при выполнении команд обработки записей, включая: номер записи, содержимое переменных, результаты выполнения команд **APPEND FROM**, **COPY**, **PACK**, **STORE** и др. Эта команда часто используется в программах для визуального контроля выполнения команд и обработки записей.

Для того чтобы при неуспешном поиске найти запись, наиболее близкую по индексному выражению, используется состояние **SET NEAR ON**. В этом случае указатель устанавливается на запись, значение индексного выражения которой следует непосредственно за искомым значением.

Пример:

```
USE ZAKUPKI INDEX Datzakup
SEEK CTOD('11/03/85')
```

Команда **FIND <литеральная строка>** аналогична команде **SEEK** и также выполняет поиск первой записи в индексированном файле БД с заданным значением индекса при установленном соответствующем главном индексе. Отличие состоит в том, что в качестве поискового значения в команде **FIND** используется не выражение, а литеральная строка. Для того чтобы задать ведущие пробелы в литеральной строке поискового значения, следует заключить эти пробелы вместе с литеральной строкой в кавычки, например **FIND** " Введение". Если литеральная строка содержится в переменной, то для задания поискового значения в таком виде следует использовать функцию макроподстановки **&**, например **FIND &Var1**, где Var1 - имя переменной. В том случае, когда в содержимое переменной должна входить литеральная строка с ведущими пробелами, нужно заключить в кавычки имя переменной вместе с символом функции макроподстановки, например "**&Var1**".

Если в поисковое значение литеральной строки входят кавычки, то тогда эту строку надо заключить в другой разделитель, например в квадратные скобки: **FIND ["Зенит"]**. Действие состояний, устанавливаемых командами **SET EXACT, SET DELETED, SET FILTER, SET NEAR**, и возвращаемые функциями **FOUND()** и **EOF()** значения полностью аналогичны команде **SEEK**.

Как уже упоминалось выше, для определения успешности поиска после команды **SEEK** часто используется функция **FOUND()**. Функция **SEEK(<выражение>[,<алиас>])** об'единяет действие команды **SEEK** и функции **FOUND()** и тем самым позволяет сократить запись такой операции. Эта функция возвращает логическое значение **.T.** (**.F.**), если найдена (не найдена) запись с заданным значением индексного выражения в текущей или заданной алиасом рабочей области. Обычно функция **SEEK()** используется в командах, определяющих ветвление процесса выполнения операций.

Пример:

```
a="Иванов"
IF SEEK(a)
   DO Proc1
ENDIF
```

Нередко после поиска требуемого значения поля нужно вывести значение другого поля этой же записи. Для выполнения такой операции можно использовать функцию **LOOKUP(<возвращаемое поле>, <поисковое выражение>, <просматриваемое поле>)**, осуществляющую поиск записи, в которой значение просматриваемого поля совпадает с поисковым выражением, и возвращающую значение из заданного поля файла БД. Возвращаемое поле может быть задано в виде любого допустимого выражения dBASE IV.

Пример:

```
USE Finans
? LOOKUP(Fam,"200",Oklad) && Выводит фамилию сотрудника
*                            с окладом 200 рублей
```

В число операторов отношений, которые формируют элементарные условия для поиска и отбора записей, входят: "больше чем","меньше чем","равно", "не равно", "больше или равно", "меньше или равно", "содержит" (см. разд. 3.1).

Кроме указанных, язык dBASE IV позволяет сформировать дополнительные операторы отношений, например "звукоподобно" и "подобно" (**LIKE**). Основой оператора "звукоподобно" является функция **SOUNDEX(<выр.C>)**, которая возвращает звукоподобный код слова. Этот код, состоящий из четырех символов, вычисляется по следующему алгоритму:

1. Оставляется первая буква заданного символьного выражения.

2. Выбрасываются буквы **a e h i o u w y** из всех позиций, кроме первой.

3. Оставшимся буквам присваивается число:

b f p v = 1
c g j k q s x z = 2
d t = 3

l = 4
m n = 5
r = 6

4. Если две или более соседние буквы имеют одно и то же число, оно отбрасывается, за исключением первой буквы.

5. Конструируется код по форме "буква цифра цифра цифра". Вместо цифр добавляются нули, если цифр меньше трех. Удаляются цифры справа, если их больше трех.

6. Формирование кода прекращается на первом небуквенном символе.

7. Пропускаются ведущие пробелы.

8. Возвращается код "0000", если первый символ, не являющийся пробелом, будет небуквенным.

Сформированный код может быть использован в качестве признака для поиска звукоподобных совпадений английских слов. Функция **DIFFERENCE(<выр.C>,<выр.C>)** определяет различие между двумя символьными строками посредством преобразования их в коды функции **SOUNDEX()** и сравнения полученных значений. Эта функция возвращает целое число в диапазоне от 0 до 4. Максимальное сходство сравниваемых символьных строк - 4, а минимальное, когда нет общих букв - 0, сходство в одной букве - 1.

Пример:

```
USE Finans
SCAN
   IF (DIFFERENCE (Fam,"Petroff") = 1
      DO Proc2
   ENDIF
ENDSCAN
```

Функция **LIKE(<образец>,<выр.C>)**, которая служит основой оператора "подобно", сравнивает строку, заданную в образце, с символьной строкой (из поля файла БД, переменной и т.д.) и возвращает **.T.**, если образец содержится в символьной строке. В образце допускается использование знаков замещения * и ?. Напомним, что первый из них замещает любое количество символов, а второй только один символ. Эти знаки не различают строчные и прописные буквы, тогда как функция **LIKE** их различает.

Чтобы избежать различения строчных и прописных букв, можно символьную строку преобразовать к прописным буквам функцией **UPPER(<выр.C>)**, а в образце задать только прописные буквы (см. разд. 3.7).

Пример:

```
? LIKE("DBASE*",UPPER("dBASE IV")) && возвращает .T.
```

При задании поискового выражения в виде содержимого символьной переменной, особенно в случае ввода целого

условия или его части с клавиатуры, могут быть ошибки, в результате чего выражение изменит свой тип, например, из логического в другой тип.

Для контроля типа выражения можно использовать функцию **TYPE(<выр.C>)**, которая определяет тип символьного выражения и возвращает следующие прописные буквы, обозначающие его тип:

C - символьная строка;
N - число;
L - логическое выражение;
M - поле **Memo**;
D - дата;
F - число с плавающей запятой;
U - неопределенное.

3.4. ВВОД И ВЫВОД ИНФОРМАЦИИ НА ЭКРАН

В предыдущем разделе уже рассматривались стандартные экранные формы ввода и вывода информации - **Browse** и **Edit**. Однако эти формы не всегда удобны для пользователя. В прикладных задачах, предназначенных для конечного пользователя, желательно улучшить оформление экрана и обеспечить максимум удобства в работе. Для этого поля файлов БД размещаются более наглядно на экране и сопровождаются поясняющей и справочной информацией, организуется система проверки вводимой с клавиатуры информации с целью избежания ошибок и т.д.

Основной командой, предназначенной для создания нестандартных экранных форм ввода и вывода информации, является

```
@<строка>,<колонка>
   [SAY <выр.>
      [PICTURE <выр.C>]
      [FUNCTION <список функций>]]
   [GET <переменная>
      [[OPEN] WINDOW <имя окна>] [PICTURE <выр.C>]
      [FUNCTION <список функций>]
      [RANGE [<нижний>] [,<верхний>]]
      [VALID <условие> [ERROR <выр.C>]]
      [WHEN <условие>] [DEFAULT <выражение>]
      [MESSAGE <выр.C>]]
      [COLOR [<стандартная область экрана>]
      [,<нестандартная область экрана>]]
```

Эта команда позволяет выводить или вводить информацию в заданном формате в указанные места экрана. Кроме того, с помощью опции **SAY** этой команды можно выводить данные на принтер или в файл на диске. Для того чтобы установить вывод

информации с помощью данной команды на экран, принтер или в текстовый файл DOS, используется команда **SET DEVICE TO SCREEN/printer/file <имя файла>**.

Строка и колонка после глагола команды позволяет задать координаты начала выводимой (вводимой) информации в виде числовых выражений. Значение **<строка>** может быть в диапазоне от 0 до максимального числа строк экрана, значение **<колонка>** - в диапазоне от 0 до 79. При выводе на принтер значение **<строка>** может находиться в интервале от 0 до 32767, а значение **<колонка>** - в интервале от 0 до 255. Если при этом координаты последующей команды **@...SAY** меньше предыдущей, то выполняется перевод листа. На некоторых принтерах эта команда не печатает до тех пор, пока не освобождается его буфер командой **EJECT** или выводом пустой строки. Если активизировано окно, то координаты **<строка>,<колонка>** отсчитываются от его левого верхнего угла. Следует отметить, что если установлено состояние **SET STATUS ON** (по умолчанию в dBASE IV устанавливается **SET STATUS OFF**, хотя в файле **Config.db** это состояние при запуске системы установлено как **ON**), то 22-я строка экрана резервируется для информации о состоянии системы. На ней высвечивается имя выполняемой команды, имя активного файла БД, номер текущей записи и их общее число, а также информация о режимах клавиатуры, если еще и установлено состояние **SET SCOREBOARD ON**. В состав этой информации входят: символ **Del**, указывающий на то, что текущая запись помечена для удаления; **Ins** - установлен режим раздвижки текста при вводе с клавиатуры; **Caps** и **Num** - установлены режимы клавиатуры, соответствующие этим клавишам. Если одновременно установлены состояния **SET STATUS OFF** и **SET SCOREBOARD ON**, то эта информация выводится на нулевую (самую верхнюю) строку экрана. Для подавления вывода информации о состоянии системы следует установить состояния **SET STATUS OFF** и **SET SCOREBOARD OFF**. В этом случае нулевую и 22-ю строки экрана можно использовать для размещения элементов создаваемой экранной формы. Если в экранной форме выводятся вычисляемые поля, то при изменении содержимого полей, которые влияют на результаты вычислений, эти вычисления выполняются, а результаты можно высветить при нажатии клавиш **PgUp** и **PgDn**. Команда **@<строка>,<колонка>** без опций очищает заданную строку, начиная с указанной колонки. Опция **SAY** выводит информацию, которую не требуется изменять. Опция **GET** выводит и позволяет обновлять и редактировать данные, содержащиеся в полях, переменных или массивах.

Собственно обновление данных в переменных **GET** (здесь и ниже под словом переменная подразумеваются также поля файла БД и элементы массива) выполняется командой **READ**, которую следует поместить после данной команды. Эта команда активизирует переменные и вводит в них значения,

находящиеся на экране. После выполнения команды **READ** экранные значения переменных **GET** удаляются и при повторном открытии формы не высвечиваются. Если использовать команду **READ SAVE**, то экранные значения переменных **GET** сохраняются и при повторном открытии экранной формы высвечиваются. Опция **COLOR** определяет цвета, используемые для высвечивания переменных **SAY** и **GET**. Участки экрана, на которых высвечиваются переменные **SAY** и **GET** относятся соответственно к стандартной и не стандартной области экрана. Коды цвета определяются так же, как и в команде **SET COLOR** (см. разд. 3.2). Опция **DEFAULT** используется для установки в переменную GET значения по умолчанию, которое должно соответствовать типу этой переменной. Это значение появляется после высвечивания экранной формы. Его можно принять клавишей **Enter** без изменения или после редакции. Выражение в этой опции вычисляется только при добавлении записей в файл БД. Следует отметить, что значение по умолчанию может быть перекрыто данными из предыдущей записи, при установленном состоянии **SET CARRY ON**.

Опция **ERROR <выр.C>** используется для высвечивания сообщения пользователя, когда не выполняется заданное в опции **VALID** условие. Это сообщение перекрывает стандартное сообщение dBASE IV **Editing condition not satisfied** (условие редактирования не выполнено), которое также высвечивается в этом случае. Опция **MESSAGE <выр.C>** высвечивает сообщение, которое появляется после перемещения курсора в поле **GET** при выполнении команды **READ**. Если установлено состояние **SET STATUS ON**, то сообщение появляется в центре нижней строки экрана, иначе оно не появляется. Эта опция временно перекрывает сообщение, выводимое командой **SET MESSAGE**. Опция **WINDOW <имя окна>** используется при установке окна для редактирования содержимого поля **Memo** переменной **GET**. Для открытия окна необходимо установить курсор в поле **Memo** и нажать клавиши **Ctrl-Home**. Клавиши **Ctrl-End** закрывают окно. Как уже ранее указывалось, если маркер поля **Memo** высвечивается прописными буквами, то оно содержит информацию, если строчными, то не содержит. При отсутствии опции **WINDOW** поле **Memo** выводится на полный экран. Для ввода и обновления текста в поле **Memo** используется текстовый редактор dBASE IV. Можно также применять любой другой редактор, для чего нужно записать его имя в качестве значения параметра **WP** в файле **Config.db**.

Опция **OPEN WINDOW <имя окна>** позволяет автоматически открыть окно для поля **Memo**. Опция **PICTURE** используется для ограничения типа вводимых данных или для форматирования выводимых данных посредством использования знаков шаблона и может включать функцию поля, которая записывается в виде символа @ и следующей за ним условной буквы, обозначающей

действие функции. Функция и шаблон поля вместе определяют ввод и вывод символов данных (более подробно см. разд. 2.5).

Отметим, что знаки шаблона и функции должны разделяться пробелами. Аргументом этой опции может быть любое символьное выражение, хотя обычно оно представляется в виде строки символов в кавычках. Если в качестве аргумента используется символьная переменная, то ее нужно заключать в круглые скобки. Функции поля можно вводить отдельно в виде списка в опции **FUNCTION**. В этом списке перечисляются (через запятую) только условные буквы без знака @. В табл. 3.3 и 3.4 приводится список функций и знаков шаблона.

Функции ^, **$**, **(**, **C**, **L**, **X** и **Z** используются с числовыми данными типа **N** и **F** (причем **(**, **C** и **X** только для вывода, т.е. с опцией **SAY**), **D** и **E** с данными типа даты, а **A**, **M**, **R** и **S<n>** с символьными данными. Можно создавать новые функции поля посредством комбинирования уже имеющихся. Например, функция **XC** выводит аббревиатуру **DB** после отрицательных чисел и **CR** после положительных. Нельзя совместно использовать взаимоисключающие функции, например, **D** и **E**.

Таблица 3.3. **Список функций поля**

Функция	Описание
!	Допускает любой символ и преобразует буквы в прописные
^	Выводит числа в формате с плавающей точкой
$	Выводит числа со знаком валюты
(	Заключает отрицательные числа в скобки
A	Допускает только алфавитные символы
B	Выравнивает текст по левому краю поля
C	Выводит аббревиатуру CR(кредит) после положительного числа
D	Устанавливает текущий формат для даты
E	Устанавливает европейский формат для даты
I	Центрирует текст внутри поля
J	Выравнивает текст по правому краю поля
L	Выводит ведущие нули чисел
M	Позволяет выбирать альтернативные значения для переменной GET
R	Высвечивает литеральные символы в поле на экране, но не вводит их в поле файла БД
S<n>	Определяет ширину выводимого поля n символами и позволяет прокручивать внутри него информацию
T	Удаляет ведущие и хвостовые пробелы из поля
X	Выводит сокращение DB (дебет) после отрицательного числа
Z	Выводит нулевое значение числа как строку пробелов

Таблица 3.4. **Перечень знаков шаблона**

Символ	Описание
!	Преобразует буквы в прописные
#	Выводит только цифры, пробелы и знаки чисел
$	Выводит символ валюты, установленный командой SET CURRENCY, вместо ведущих нулей
*	Выводит знаки * вместо ведущих нулей
,	Отмечает каждые три порядка слева от десятичной точки
.	Определяет положение десятичной точки
9	Допускает вывод только цифр для символьных данных. Для числовых данных выводит цифры и знаки чисел
A	Допускает вывод только букв
L	Допускает вывод только логических данных
N	Допускает вывод букв и цифр
X	Допускает вывод любого символа
Y	Выводит только логические значения Y,y,N,n. Преобразует y,n в прописные

Функция **$** выводит символ валюты до или после значения переменной **GET**. Символ валюты размером до 9 символов и его размещение справа или слева от числа устанавливаются командами **SET CURRENCY TO <выр. C>** и **SET CURRENCY LEFT/RIGHT**.

Для просмотра длинного значения символьной строки в выводимом на экране более коротком поле используется функция **S<n>**, где **n** - количество позиций этого поля. Число **n** не должно отделяться от символа **S** пробелами. Этот просмотр выполняется навигационными клавишами **Вправо**, **Влево**, **Home** и **End**. Одновременно с ним можно редактировать символьную строку. Функция **M** позволяет для переменной **GET** выбирать клавишей **Пробел** заданные в списке альтернативные значения (в виде литералов, разделяемых запятой), например **FUNCTION "M <список значений>"**.

Можно также выбирать эти значения, нажимая на клавиатуре их первую букву. Функции **B,I** и **J** выравнивают значения полей. По умолчанию строки выравниваются влево, а числа вправо.

Знаки шаблона !, #, **9**, **A**, **N** и **X** используются в опциях **SAY** и **GET**. Все эти знаки, кроме !, не разрешают ввод недопустимых символов, но позволяют их высвечивать. Следует напомнить, что при выводе чисел в состав шаблона нужно учитывать знаки для десятичной точки и знака числа (если требуется). Если в качестве знака шаблона используется знак, отличный от приведенных в таблице, то этот знак высвечивается в поле переменной на экране и вводится в нее (для переменной **GET**). Если вместе с шаблоном используется функция **R** (для символьных переменных), то не предусмотренные в шаблоне знаки высвечиваются в поле, но не вводятся в переменную.

Опция **RANGE** используется для определения нижних и верхних границ вводимых значений символьных и числовых данных и данных типа даты. Можно вместо двух указывать только одну из границ, для чего нужно поставить запятую до или после значения границы. При наборе значения, выходящего за рамки указанных границ, в строке сообщений при состоянии **SET STATUS ON** высвечивается предупреждение в виде заданных границ: **RANGE is <m> to <n>** - двусторонние границы, **Lowerbound is <n>** - правая граница, **Upper bound is <m>** - левая граница. Если установлено состояние **SET STATUS OFF**, то эти границы высвечиваются на нулевой строке. Ввод неисправленного значения можно выполнить клавишей **Enter**.

Опция **VALID** устанавливает условие, которое должно быть выполнено перед тем, как данные вводятся в переменную **GET**. Если условие не выполнено, высвечивается сообщение **Editing condition not satisfied** (условие редактирования не выполнено) или же сообщение, установленное опцией **ERROR**. Нажатие клавиши **Enter** отменяет проверку этого условия. В качестве условия можно, например, записать вызов функции, определенной пользователем, если эта функция возвращает логическое значение, а также и вызов процедуры.

Предыдущая опция проверяет условия ввода данных в переменную, тогда как опция **WHEN** позволяет задать условия для входа курсора в переменную **GET**. Если в результате проверки этого условия возвращается логическое значение **.F.**, то курсор пропускает это поле и перемещается в следующее. В случае возврата **.T.** курсор остается в поле и разрешается вводить в него данные.

Перехват и обработку любой ошибочной ситуации при выполнении полноэкранных операций по вводу информации, включая ввод данных, не соответствующих типу поля, данных вне заданного диапазона и т.д., можно также выполнить посредством установки состояния **ON READERROR [<команда>]**, которое активизирует команду, программу или процедуру после возникновения ошибки. Команда **ON READERROR** без аргумента отключает состояние перехвата ошибки. В этой команде запрещается использование ряда команд (тех, что и при построении функций пользователя - см. разд. 3.6).

Разрабатываемая экранная форма может размещаться на нескольких страницах экрана (до 32). Передвижение по ним осуществляется клавишами **PgDn** и **PgUp**. Для организации многостраничной экранной формы надо по завершении каждой страницы поставить команду **READ**. Для того чтобы использовать такую экранную форму, нужно предварительно записать ее в файл экранных форм с расширением **.fmt**. Открытие этой формы выполняется командой **SET FORMAT TO <имя .fmt файла>**. Экранная форма, открытая данной командой, используется вместо стандартного экрана в полноэкранных командах **EDIT** и **APPEND**.

Для создания экранной формы можно также использовать команду **CREATE/MODIFY SCREEN** или окно **Forms** в УЦ (см. разд. 2.5).

Пример:

```
USE FIO
CLEAR
@ 10,5 SAY TRIM(FAM)+" "+NAME
```

В этом примере выводится на экран через пробел поле фамилии (FAM) и имени (NAME) первой записи файла БД FIO в 10-й строке с 5-й позиции с удалением ведущих пробелов из поля FAM.

Пример:

```
day="   "
@ 1,1 SAY "Выбрать день недели" GET day ;
     PICTURE "@M Пон,Втр,Срд,Чтв,Птн,Суб,Вск";
     MESSAGE "Для выбора дня недели нажмите клавишу;
              <Пробел>, а для ввода - <Enter>"
READ
```

В первой строке предыдущего примера на экран с первой позиции выводится сообщение "Выбрать день недели". Выбор значения переменной day осуществляется с помощью функции **@M** нажатием клавиши **Пробел**. Ввод выбранного значения выполняется **Enter**.

Пример:

```
STORE SPACE (25) TO mname
@ 10,10 SAY "Введите имя: " GET mname
READ
```

Для обеспечения пользователю удобства работы с экранной формой часто кроме пояснительных надписей на самом экране требуется организовать автоматический вызов на экран справочной информации по текущему обрабатываемому полю (контекстно-чувствительная справка). Чтобы организовать такую справочную систему в программе, нужно знать, на каком поле в данный момент находится курсор. Это можно сделать с помощью функции **VARREAD()**, которая возвращает имя поля или переменной, обрабатываемой в текущий момент. Ниже приводится процедура для организации контекстно-чувствительной справки, которая выводится клавишей **F1** в заданное окно на экране.

```
ON KEY F1 DO Spravka
@ 10,10 SAY "Склад N 1" GET Sklad1
@ 12,10 SAY "Склад N 2" GET Sklad2
READ
PROCEDURE Spravka
clr = INKEY()  && Очистка буфера клавиатуры
```

```
ACTIVATE WINDOW Winspr
DO CASE
   CASE = VARREAD() = "Sklad1"
   ? "Описание профиля склада N 1"
   CASE = VARREAD() = "Sklad2"
   ? "Описание профиля склада N 1"
ENDCASE
? INKEY(0) && Удержание экрана
DEACTIVATE Winspr
RETURN
```

Здесь для перехвата нажатой клавиши **F1** используется команда

**ON ERROR <команда> /ESCAPE <команда>
/KEY [LABEL<имя клавиши>] [<команда>]**

Эта команда устанавливает состояние **ON** перехвата ошибки (опция **ERROR**) при выполнении команд, обнаружении нажатия клавиши **Esc** (опция **ESCAPE**) или любой клавиши (опция **KEY** без ключевого слова **LABEL**), а также заданной клавиши или комбинации клавиш (с ключевым словом **LABEL**). После обнаружения состояние **ON** выполняется указанная команда, вызывается функция, определенная пользователем, или процедура. Следует отметить, что функция или процедура должны быть предварительно откомпилирована, так как при их вызове компиляция не производится. Состояние **ON** отменяется командой **ON ERROR/ESCAPE/KEY** без указания выполняемой команды или после выхода из dBASE IV. Команда **ON ESCAPE** не работает, если установлено состояние **SET ESCAPE OFF**.

После обнаружения состояния **ON** и выполнения заданной команды это состояние временно отключается до завершения выполнения команды или процедуры. Однако внутри процедуры можно установить новое состояние **ON** и, таким образом, образовать вложенные состояния **ON**. Если одновременно установлены все состояния, предусмотренные данной командой, то их обнаружение выполняется в следующей последовательности: **ON ERROR**, **ON ESCAPE** и **ON KEY**. Если, например, **ON KEY** и **ON ESCAPE** установлены одновременно, нажатие клавиши **Esc** вызывает исполнение команды, указанной в **ON ESCAPE**. Команда **ON KEY** в этом случае игнорируется. Если действует только команда **ON KEY** и нажата любая, отличная от **Esc** клавиша, то команда, указанная в **ON KEY**, выполняется после завершения текущей команды. Например, нажатие любой клавиши при выполнении индексирования файла с помощью команды **INDEX** в состоянии **ON KEY** не прерывает процесса индексирования. Если в буфере сохраняется значение клавиши, то команда, заданная в **ON KEY**, выполняется повторно. Следовательно, для правильного использования команды **ON KEY** нужно своевременно удалять из буфера сохраненное значение клавиши

с помощью функции **INKEY()** или команды **READ**. Команда **ON ERROR** обрабатывает только ошибки DBASE IV, включая синтаксические ошибки, ошибки вычислений и ввода данных и др., но не ошибки на уровне операционной системы, например вследствие недоступности принтера или диска.

В качестве команды, выполняемой после перехвата клавиши в состоянии **ON KEY**, нельзя использовать те же команды, что и при создании функции, определенной пользователем (см. команду **FUNCTION** в разд. 3.6).

Команда **ON KEY LABEL <имя клавиши>** позволяет перехватить определенную нажатую клавишу. Имя этой клавиши можно задать неявно, поместив в переменную и использовав ее вместе с функцией макроподстановки **&**. Команда **ON KEY LABEL** без аргумента устанавливает перехват любой клавиши, как и команда **ON KEY**. Выполнение команды **ON KEY** без опции после команды **ON KEY LABEL** без аргумента отключает перехват клавишей. Состояние **ON KEY** вызывает перехват клавишей во время выполнения команд **BROWSE**, **EDIT**, **READ**, меню и после выполнения всех команд. Такие команды, как **LIST**, **SORT** и **INDEX**, не прерываются до их завершения. Можно перехватывать любые клавиши, а также парные комбинации клавиш **Ctrl**, **Shift**, **Alt** с функциональными клавишами, цифрами и буквами, а клавиши **Ctrl** также и с навигационными клавишами, например, **Ctrl-F1**, **Alt-0**, **Shift-F9**, **Ctrl-PgUp**, **Ctrl-PgDn** и т.д. В качестве имен клавиш и их комбинаций выступает обычная запись, которая встречается везде по тексту. Исключение составляют навигационные клавиши, имена которых должны быть представлены английскими словами: **Вправо - Rightarrow**, **Влево - Leftarrow**, **Вверх - Uparrow**, **Вниз - Dnarrow**. Слова можно писать прописными и строчными буквами или их комбинацией. Если клавиша запрограммирована командой **SET FUNCTION** и для нее установлен перехват командой **ON KEY**, то последняя имеет приоритет. Если команда **ON KEY** перехватывает клавишу при вводе ее в переменную **GET**, выполняется заданная в **ON KEY** команда и управление возвращается в точку перед этой командой. При этом перехваченный символ клавиши не помещается в переменную **GET**. Если в состоянии перехвата клавиши командой **ON KEY** активна команда **WAIT TO <переменная>**, последняя имеет приоритет и выполняется независимо от команды **ON KEY**, причем перехватываемая клавиша заносится в переменную команды **WAIT**.

Пример:

```
ON ERROR DO Procerr
PROCEDURE Procerr
        .
        .
        .
```

```
ON ERROR
RETURN
ON KEY DO Stop
PROCEDURE Stop
       .
       .
       .
WAIT "Нажмите клавишу A для прекращения печати"+;
     " или любую клавишу для продолжения" TO Otvet
IF UPPER(Otvet) = "A"
   RETURN TO MASTER
ENDIF
RETURN
```

В примере, приведенном выше, для приостановки процесса последовательного выполнения программы используется команда

WAIT [<сообщение>] [TO <переменная>]

которая вызывает паузу до нажатия любой клавиши и высвечивает в нижней строке экрана сообщение. После нажатия клавиши выполнение программы продолжается, а в переменную заносится код ASCII этой клавиши (для клавиши **Enter** заносится нулевой код ASCII, а для непечатаемых клавиш - графический символ соответствующего кода). Сообщение представляет собой символьное выражение. Если сообщение не указано, то высвечивается стандартное сообщение: **Press any key for continue...** (нажмите любую клавишу для продолжения).

Пример:

```
WAIT "Будете продолжать? (Y/N)" TO Mcon
IF UPPER (Mcon) = 'Y'
   RETURN
ENDIF
```

Часто при обработке ошибочной ситуации в процедуре (программе), вызываемой командой **ON ERROR DO <имя программы>**, используются команда **RETRY**, которая осуществляет повторное выполнение последовательности команд, вызвавших ошибку. Обычно в этой процедуре предусматриваются средства локализации и исправления ошибки. Следует однако учесть, что эта команда не выполняет повторно команд **IF** или **DO WHILE**.

При использовании в программном файле команда **RETRY** закрывает этот файл и очищает значение, возвращаемое функцией **ERROR()**. Эта функция возвращает номер ошибки, указанной в перечне сообщений об ошибках. Текст сообщения об ошибке можно получить, используя функцию **MESSAGE()**.

Пример:

```
* Главная программа Main
ON ERROR DO Curl WITH ERROR()
USE FILE1
```

```
RETURN
* Программа Cur1
PARAMETERS Er
IF Er = 3
   CLOSE DATABASES
ENDIF
RETRY
RETURN
```

В приведенном примере в случае, если открывается уже открытый файл (ошибка с номером 3), вызывается процедура, в которой закрываются все файлы БД и затем осуществляется возврат на команду открытия файла, вызвавшую ошибочную ситуацию.

Для отслеживания нажатых клавиш используются функции **INKEY([n])** и **LASTKEY()**. Числовой аргумент функции **INKEY([n])** позволяет приостановить выполнение программы на заданное количество секунд. Если не нажата никакая клавиша, то возвращается ноль. Функция **INKEY()** вместе с возвращением кода клавиши удаляет ее значение из буфера клавиатуры.

В табл. 3.5 представлены десятичные коды клавиш и их комбинаций. Функции **INKEY([n])** и **LASTKEY()** используются в программах для перехвата нажатых клавиш с целью выполнения определенных действий. В принципе, с их помощью можно реализовать те же функции, что и командой **ON KEY**.

Ниже приводится программа, в которой с помощью функции **LASTKEY()** перехватывается клавиша, назначенная для вывода на экран информации.

Пример:

```
SET FUNCTION 9 TO CHR(16) && 16 - код ASCII клавиши Ctrl-P
DO WHILE .T.
   * Ввод данных в GET переменные
   . . .
   @ 19,40 SAY "Для вывода данных нажмите F9"
   READ
   IF LASTKEY() = -8
      DO Scrform
         LOOP
      ELSE
         EXIT
      ENDIF
   ENDDO
```

Здесь используется функция **CHR()**, которая возвращает символ, соответствующий его десятичному коду ASCII (более подробно см. разд. 3.7). При выполнении программы обработки ошибок с помощью команды **ON ERROR DO <имя программы>** в начале этой программы целесообразно отключить буфер клавиатуры. Тем самым отключается состояние **ON KEY** и

действие функции **INKEY()**, что позволяет без помех обработать ошибочную ситуацию.

Таблица 3.5. **Десятичные ASCII коды клавиш**

Клавиша	Код	Клавиша	Код
Ctrl-A	1	F1	28
Ctrl-B	2	Ctrl-Home	29
Ctrl-C	3	Ctrl-PgDn	30
Ctrl-D	4	Ctrl-PgUp	31
Ctrl-E	5	Пробел	32
Ctrl-F	6	Backtab	-400
Ctrl-G	7	Ctrl-Backspace	-401
Ctrl-H	8	Ctrl-Enter	-402
Ctrl-I	9	F2	-1
Ctrl-J	10	F3	-2
Ctrl-K	11	F4	-3
Ctrl-L	12	F5	-4
Ctrl-M	13	F6	-5
Ctrl-N	14	F7	-6
Ctrl-O	15	F8	-7
Ctrl-P	16	F9	-8
Ctrl-Q	17	F10	-9
Ctrl-R	18	Ctrl-F1	-10
Ctrl-S	19	Ctrl-F2	-11
Ctrl-T	20	Ctrl-F3	-12
Ctrl-U	21	Ctrl-F4	-13
Ctrl-V	22	Ctrl-F5	-14
Ctrl-W	23	Ctrl-F6	-15
Ctrl-X	24	Ctrl-F7	-16
Ctrl-Y	25	Ctrl-F8	-17
Ctrl-Z	26	Ctrl-F9	-18
Esc	27	Ctrl-F10	-19
Shift-F1	-20	Alt-n	-422
Shift-F2	-21	Alt-o	-421
Shift-F3	-22	Alt-p	-420
Shift-F4	-23	Alt-q	-419
Shift-F5	-24	Alt-r	-418
Shift-F6	-25	Alt-s	-417
Shift-F7	-26	Alt-t	-416
Shift-F8	-27	Alt-u	-415
Shift-F9	-28	Alt-v	-414
Shift-F10	-29	Alt-w	-413
Alt-a	-435	Alt-x	-412
Alt-b	-434	Alt-y	-411
Alt-c	-433	Alt-z	-410
Alt-d	-432	Alt-1	-451
Alt-e	-431	Alt-2	-450
Alt-f	-430	Alt-3	-449
Alt-g	-429	Alt-4	-448
Alt-h	-428	Alt-5	-447
Alt-i	-427	Alt-6	-446
Alt-j	-426	Alt-7	-445
Alt-k	-425	Alt-8	-444
Alt-l	-424	Alt-9	-443
Alt-m	-423	Alt-0	-452

Отключить буфер можно, установив его размер равным нулю. Для этого используется команда **SET TYPEAHEAD TO <выр.N>**, определяющая размер буфера клавиатуры (от 0 до 32000 символов, по умолчанию 20), который также влияет на скорость ввода данных. Для увеличения скорости ввода данных нужно установить достаточно большой размер буфера, так как при определенной скорости ввода в буфере может накопится больше символов, чем его размер. При этом не помещающиеся в буфере символы теряются, что сигнализируется звуком, если **SET BELL ON**.

Команда **SET TYPEAHEAD TO** действует только в состоянии **SET ESCAPE ON**. В командах **EDIT** и **APPEND** буфер равен 20 символам, независимо от установки команды **SET TYPEAHEAD**. Описанная выше команда **ON KEY** позволяет перехватить нажатую клавишу или их комбинацию и выполнить ряд операций. Таким образом, клавише или их комбинации можно назначить выполнение определенных, специфических для конкретной прикладной программы действий. Для этой цели часто используют функциональные клавиши и их комбинации. Назначение клавише определенных действий называется ее программированием. Специально для программирования функциональных клавиш можно использовать команду **SET FUNCTION <выр.N>/<выр.C>/<функциональная клавиша> TO <выр.C>**, которая позволяет поставить в соответствие функциональным клавишам **F2-F10**, комбинациям клавиш с **Shift-F1** по **Shift-F9** и с **Ctrl-F1** по **Ctrl-F10** строку длиной до 238 символов. Клавиша **F1** не программируется и всегда используется только для вывода справочной информации.

Функциональные клавиши можно задавать в виде их номера (см. ниже), символьного выражения или имени. Присвоенная клавише или их комбинации строка при их нажатии будет передаваться в соответствии с текущей операцией ввода. Если в конце строки поставить точку с запятой, то она будет интерпретироваться и выполняться как команда. В строку может входить несколько команд, в конце каждой из которых должна стоять точка с запятой. Клавиши **Shift-F10**, все комбинации функциональных клавиш с клавишей **Alt** используются для вызова макрокоманд, поэтому их нельзя программировать.

Команда **SET FUNCTION** позволяет задавать функциональную клавишу явно или в виде числового или символьного выражения. Выполняющая аналогичные функции команда **ON KEY** имеет приоритет перед командой **SET FUNCTION**. Если эти команды присваивают различные командные строки одной клавише, то выполняется строка, присвоенная командой **ON KEY**.

Функциональные клавиши **F1-F10** имеют стандартные назначения в виде команд, представленных в табл. 3.6. Назначения функциональных клавиш можно также установить с помощью меню команды **SET**, а также в файле **Config.db**.

Таблица 3.6. **Стандартные назначения функциональных клавиш**

Клавиша	Команда	Клавиша	Команда
F1	help;	F6	display status;
F2	assist;	F7	display memory;
F3	list;	F8	display;
F4	dir;	F9	append;
F5	display structure;	F10	edit;

Всего можно программировать 9 клавишей и 19 их комбинаций. Запрограммированные клавиши и их комбинации нумеруются числами от 1 до 28, причем 1 присваивается клавише **F2** и т.д. в порядке их программирования.

Примеры:

```
SET FUNCTION F10 TO "SET;"
* При нахатии клавиши F10 выполняется команда SET
SET FUNCTION F9 TO "CLEAR;USE Fio;LIST;"
* Выполняет последовательность команд CLEAR, USE, LIST
```

Функция **FKLABEL (<выр.N>)** позволяет определить клавишу или их комбинацию, которая имеет заданный номер. Эту функцию можно использовать в прикладных программах при переназначении функциональных клавиш в зависимости от решаемой задачи. Кроме стандартной клавитуры IBM существуют другие, в которых может быть иное число программируемых клавиш. Для возвращения общего числа таких клавиш используется функция **FKMAX()**.

В программах при выполнении операций с полноэкранными командами типа **BROWSE**, **EDIT**, **APPEND**, **CNANGE**, **CREATE**, **MODIFY**, **@...SAY...GET** и **READ** нередко требуется определить, были ли внесены изменения в файл БД. Для этого используется команда **READKEY()**, которая возвращает код (целое число), определяющий клавишу, нажатую перед выходом из полноэкранной команды. Это число зависит от того, обновлялись или остались неизменными данные при выполнении команды. В табл. 3.7 приведены коды клавиш выхода из полноэкранных команд.

Пример:

```
IF READKEY()>=256    && Данные обновлены
   REPLACE Oklad WITH Sum
ENDIF
```

В гл. 2 были описаны действия по созданию экранных форм ввода и вывода данных с помощью меню УЦ. Данная функция реализуется с помощью специальной полноэкранной команды **CREATE/MODIFY SCREEN <имя файла>/?**, которая передает управление в окно **Forms** УЦ для разработки (изменения) экранных форм ввода-вывода информации и позволяет создать (изменить) файл экранной формы.

Таблица 3.7. **Возвращаемые коды клавиш**

Без обнов-ления	После обнов-ления	Клавиша	Назначение
0	256	Ctrl-S,Ctrl-H,	Назад на один символ
-	256	Backspace	Назад на один символ
1	257	Ctrl-D,Ctrl-L,	Вперед на один символ
4	260	Ctrl-E,Ctrl-K,	Назад на одно поле
5	261	Ctrl-J,Ctrl-X,	Вперед на одно поле
6	262	Ctrl-R,PgUp	Назад на один экран
7	263	Ctrl-C,PgDn	Вперед на один экран
12	-	Ctrl-Q, Esc	Завершить без сохранения
-	270	Ctrl-W,Ctrl-End	Завершить с сохранением
15	271	Enter,Ctrl-M	Ввод или заполнение последней записи
16	-	Enter,Ctrl-M	В начало записи при добавлении
33	289	Ctrl-Home	Переключение меню
34	290	Ctrl-PgUp	Выход из полного экрана
35	291	Ctrl-PgDn	Вход в полный экран
36	292	F1	Клавиша помощи

При разработке экранной формы сначала создается файл разработки экранной формы с расширением **.scr**. На основе этого файла генерируется программный файл с расширением **.fmt**, содержащий команды **@...SAY... GET** и **READ**, обеспечивающие ввод и вывод данных и другие действия при работе с экранной формой.

Файл разработки с расширением **.scr** предназначен не только для генерации файла **.fmt**, но и для последующей модификации экранной формы. Если файл **.scr** уничтожен, то тогда уже нельзя модифицировать файл экранной формы с помощью команды **CREATE/MODIFY SCREEN**. Однако всегда остается возможность непосредствено модифицировать программный файл экранной формы с расширением **.fmt** с помощью текстового редактора dBASE IV, вызываемого командой **MODIFY COMMAND**. Можно также использовать любой другой текстовой редактор. После первого открытия файла экранной формы типа **.fmt** на его основе псевдокомпилятором dBASE IV создается файл с расширением **.fmo** (объектный модуль), который предназначен для непосредственного выполнения. В программах файл экранной формы можно открыть командой **SET FORMAT TO <имя файла экранной формы>)** и закрыть командой **SET FORMAT TO** как обычный файл, что позволяет выполнить обработку файла БД в соответствующей экранной форме. Опция **?** команды **CREATE/MODIFY SCREEN**, альтернативная заданию имени файла экранной формы, позволяет, если открыт каталог, высветить на экране список файлов типа **.scr**, связанных с активным файлом БД или выборкой. Если каталог закрыт, то эта опция высвечивает список

файлов **.scr** на текущем диске. После появления списка файлов **.scr** на экране можно выбрать один из них для модификации.

В dBASE IV можно также использовать файлы экранных форм, созданные в dBASE III+. После открытия такой файл преобразуется в формат dBASE IV и ему присваивается расширение **.sc3**. Файлы экранных форм, разработанные в dBASE IV, нельзя использовать в dBASE III+ (более подробно о разработке экранных форм ввода и вывода данных с помощью режима УЦ, управляемого меню, см. в разд. 2.5).

В программах обработки файлов, когда используются несколько экранных форм ввода и вывода информации, нередко после вызова формы требуется очистить или закрасить определенную область экрана. Очистку можно сделать командой

@ <строка1>,<колонка1> CLEAR
[TO <строка2>,<колонка2>]

которая удаляет содержимое прямоугольной области экрана, определяемую координатами верхнего левого - **<строка1>, <колонка1>** и правого нижнего - **<строка2>,<колонка2>** - углов прямоугольника. Если не указаны координаты правого нижнего угла, то очищается строка, начиная с позиции **<колонка1>**.

Пример:

```
@ 2,10 CLEAR TO 15,79
```

Для закрашивания части экрана используется команда

@ <строка1>,<колонка1> FILL TO <строка2>,<колонка2>
[COLOR <код цвета>]

Эта команда закрашивает прямоугольную область экрана, определяемую координатами верхнего левого - **<строка1>, <колонка1>** и правого нижнего - **<строка2>,<колонка2>** - углов прямоугольника. Коды цвета приведены при описании команды **SET COLOR**.

Пример:

```
@ 2,10 FILL TO 15,79 COLOR R/B
```

С целью улучшения восприятия информации на экране можно использовать команду

@ <строка1>,<колонка1> TO <строка2>,<колонка2>
[DOUBLE/PANEL/<символы определения рамки>]
[COLOR <код цвета>]

которая рисует на экране прямоугольную рамку, определяемую координатами верхнего левого - **<строка1>,<колонка1>** - и правого нижнего - **<строка2>,<колонка2>** - углов рамки. Опция **DOUBLE** позволяет рисовать двойную линию рамки (по умолчанию рисуется одинарная линия), опция **PANEL** - рамку в

виде сплошной полосы. Кроме того, имеется возможность рисовать линию рамки заданными символами. Для этого задается список символов для определения рамки. Можно задать различные символы для сторон и углов рамки. В этом случае символы разделяются запятыми и располагаются в следующем порядке: **t**, **b**, **l**, **r**, **tl**, **tr**, **bl**, **br**, где **t** - верх; **b** - низ; **l** - левая сторона; **r** - правая сторона; **tl** - левый верхний угол; **tr** - правый верхний угол; **bl** - левый нижний угол; **br** - правый нижний угол.

Пример:

```
@ 5,13 TO 10,40 DOUBLE COLOR R/BG
```

Для полной очистки всего экрана используется команда

CLEAR [ALL/FIELDS/GETS/MEMORY/MENUS/ POPUPS/TYPEAHEAD/WINDOWS]

Она очищает экран, устанавливает курсор в левый нижний угол экрана и освобождает все незаконченные опции **GET** команды **@...SAY...GET**. Различные опции этой команды также закрывают файлы БД, очищают переменные, списки полей, окна, горизонтальные и вертикальные меню и буфер клавиатуры.

Опция **ALL** закрывает все открытые файлы БД, горизонтальные и вертикальные меню, очищает все переменные (за исключением системных переменных), элементы массивов и выбирает рабочую область 1. Одновременно закрываются все связанные индексные файлы (**.ndx** и **.mdx**), файлы экранных форм (**.fmt**) и файлы **Memo** (**.dbt**), а также файл каталога (**.cat**).

Опция **FIELDS** освобождает список полей, созданный командой **SET FIELDS**. В отличие от команды SET FIELDS TO, которая удаляет активные поля файла БД из списка полей, данная команда освобождает все поля, в том числе включенные из других рабочих областей. Выполнение этой опции автоматически влечет выполнение команды **SET FIELDS OFF**. Опция **GETS** освобождает все команды **@...GET**, следующие после последней выполненной команды **CLEAR ALL**, **CLEAR GETS** или **READ**. По умолчанию разрешается использовать до 128 опций **GET** перед командами **CLEAR GETS** и **READ**. Опция **MEMORY** освобождает все переменные (за исключением системных) и элементы массивов. Опция **MENUS** очищает все горизонтальные меню, опция **POPUPS** все вертикальные меню, а опция **WINDOWS** все окна с экрана и удаляет их из памяти. Опция **TYPEAHEAD** - освобождает буфер клавиатуры.

Команда **@...SAY...GET** является универсальной для организации ввода и вывода информации на экране. В программах нередко требуется вводить с клавиатуры данные для обеспечения процесса обработки информации, например имя файла БД, данные для ветвления программы, поисковые значения и т.д. Для этого предусмотрена специальная команда

ACCEPT [<сообщение>] TO <переменная>

которая позволяет вводить с клавиатуры данные символьного типа в переменную. Перед полем переменной можно высветить сообщение, например о характере вводимых данных. Это сообщение представляет собой символьную строку (должна заключаться в двойные, одинарные кавычки или квадратные скобки), переменную или любое символьное выражение длиной до 254 позиций. После набора данных с клавиатуры для ввода их в переменную следует нажать клавишу **Enter**.

Пример:

```
ACCEPT "Введите имя:" TO name
```

Кроме этой команды для ввода данных с клавиатуры можно использовать команду

INPUT [<сообщение>] TO <переменная>

которая выполняет аналогичные функции, но позволяет вводить в переменную с клавиатуры данные не только символьного, но и других типов, а также в виде заданного выражения, что не допускается в команде **ACCEPT**. Тип выражения определяет тип переменной, в которую вводится значение. После набора выражения с клавиатуры для ввода его в переменную надо нажать клавишу **Enter**.

Следует напомнить, что вводимые данные типа даты должны заключаться в фигурные скобки. Если же дата вводится без этих скобок (т.е. в виде символьной строки), то для преобразования в данные типа даты следует использовать функцию **CTOD()** (более подробно см. разд. 3.7).

При разработке экранных форм весьма полезными являются функции **COL()** и **ROW()**, которые возвращают текущий номер колонки и строки расположения курсора. Эти функции можно использовать для относительной адресации позиций экрана в команде **@...SAY...GET**, например @ 5,COL()... Вместо функции **COL()** для этой цели можно использовать специальный оператор **$**: @ 5,$, но функция **COL()** работает быстрее. С помощью относительной адресации можно, например, выводить информацию в различные места на экране с помощью одной процедуры. Фрагмент такой процедуры может включать:

```
ROW() = nrow
COL() = ncol
DO WHILE COL() < ecol
   @ ROW(),COL() + 5 SAY "I"
ENDDO
```

Как уже указывалось выше, программирование клавиш можно выполнить не только командой **SET FUNCTION**, но и командой **SET**. Это полноэкранная управляемая с помощью меню команда предназначена для высвечивания и изменения значений многих

команд **SET**. При выполнении этой команды высвечивается меню, состоящее из пяти позиций:

Options - высвечивает значения, установленные по умолчанию для многих команд **SET**. Эти значения можно изменить с клавиатуры;

Display - позволят изменить цвет и интенсивность высвечиваемого текста, заголовков, строки состояния и полей. Сохранение изменений выполняется клавишами **Ctrl-End**. Изменения сохраняются в течение текущего сеанса. При выходе из dBASE IV все цвета (кроме интенсивности свечения) возвращаются в исходные;

Keys - позволяет программировать клавиши и их комбинации;

Disk - позволяет изменять заданный по умолчанию диск и маршрут по директориям;

Files - позволяет задавать: альтернативные файлы, файлы экранных форм, тип выходного устройства и файлы индексов.

В программах нередко требуется определить текущее состояние команд **SET**. Для этого можно использовать функцию **SET(<выр.C>)**, которая возвращает целое число или значение **ON** или **OFF** команд **SET**.

Символьное выражение **<выр.C>** представляет собой ключевое слово команды. Если в команде **SET** устанавливается целое числовое значение и значения **ON** и **OFF**, то возвращается только **ON** или **OFF**.

Примеры:

```
? SET("BELL")       && Высвечивает ON
? SET ("CARRY")     && Высвечивает OFF
```

3.5. СОЗДАНИЕ ОТЧЕТОВ И МАРОК

Команда **@...SAY...GET**, описанная в предыдущем разделе, предназначена для ввода информации с клавиатуры в поля файла БД, массивы и переменные и вывода ее на экран.

Опция **SAY** позволяет также выводить эту информацию и на принтер. Кроме этой универсальной команды существует команда, специально предназначенная для вывода данных на экран и принтер:

?/?? [<выр.1> [PICTURE <выр.C>]
[FUNCTION <список функций>] [AT <выр.N>]
[STYLE <номер шрифта>]] [,<выр.2>...] [,]

которая выводит значение одного или более заданных символьных выражений. Если установлено состояние **SET PRINT ON**, то выполняется вывод на принтер, если **SET PRINT OFF**, то на экран. Эта команда в форме **?** выводит каждое выражение на следую-

щей строке, а в форме ?? все выражения на одной строке. Опции **PICTURE** и **FUNCTION**, так же, как и в команде @...**SAY... GET**, позволяют форматировать вывод.

Для форматирования используются шаблоны и функции поля. В качестве знаков шаблонов используются те же, что описаны в команде @...**SAY... GET**.

Из числа функций поля последней команды в данной используются только шесть, а две функции введены новые. Всего в данной команде предусмотрено восемь функций поля, которые обозначаются буквами **H, V, B, I, J, T, $, L**. Последние шесть описаны в команде @...**SAY...GET**.

Функция **@H<n>** позволяет делать горизонтальную, а **@V<n>** - вертикальную развертку поля данных, где **n** - максимальное количество колонок или строк выводимого поля. Эти функции по-разному обрабатывают длинные и короткие поля. Длинными считаются поля, длина которых превышает шаблон опции **PICTURE** команды ?/??, а короткими - поля, которые не заполняют целиком этот шаблон.

Функция **@H** развертывает длинные поля горизонтально вправо и вниз, раздвигая вправо и влево короткие поля и другой текст по соседству, если не присутствует опция **AT**, в которой задается позиция левой колонки начала вывода.

Для горизонтальной развертки должно быть установлено значение **.T.** системной переменной **_wrap** (более подробно о системных переменных см. ниже в этом разделе).

Ширина колонки при выводе поля **Memo** устанавливается командой **SET MEMOWIDTH TO <выр.N>** (по умолчанию 50 позиций). Ее можно также установить в файле **Config.db**. Максимальное значение ширины колонки 32000, минимальное 8.

При выводе на печать поля **Memo** в случае использования функции шаблона **@H** его содержимое развертывается внутри границ, определенных системными переменными **_lmargin** и **_rmargin**, а установленная командой **SET MEMOWIDTH** ширина поля игнорируется.

Функция **@V<n>** развертывает длинные поля вертикально вниз в колонке заданной ширины. Эта функция может использоваться для развертки по вертикали и поля **Memo**. Если **n = 0**, то поле **Memo** развертывается в пределах ширины, установленной командой **SET MEMOWIDTH**.

При совместном использовании функции **@V** и шаблона **PICTURE** для вывода поля **Memo** его текст может быть обрезан в конце, если длина строки больше, чем установлено в шаблоне.

Функции **@B**, **@I** или **@J** перекрывают установку системной переменной **_alignment** и позиционирование текста в поле **Memo**. Если в конце команды ?/?? стоит запятая, то при вертикальной развертке сначала печатается только первая строка до тех пор, пока не встретятся команды **RETURN, ENDPRINTJOB** или команда

?/?? без запятой в конце. Это позволяет выводить ряд полей в одной строке с помощью нескольких команд ?? и управлять выводом второй и последующих строк полей при их вертикальной развертке.

Опция **STYLE** предназначена для вывода на печать текста различными шрифтами, например жирным или наклонным. Номер шрифта обозначается условными буквами, цифрами или их комбинацией. В качестве условных букв используются: **B** - жирный, **I** - наклонный, **U** - с подчеркиванием, **R** - приподнятый, **L** - опущенный. Можно вместо букв задавать цифры от 1 до 5, которые соответствуют шрифтам, предварительно определенным в файле **Config.db** с помощью параметра **PRINTER** (см. о файле **Config.db** в разд. 3.1).

Опция **STYLE** позволяет, комбинируя шрифты, печатать текст на одной строке в различном стиле. Для изменения типа шрифта можно также использовать команду ??? и системные переменные **_pscode** и **_pecode**.

Команда ? без опций выводит одну пустую строку. Для пропуска одной или нескольких строк можно также использовать системную переменную **_pspacing**.

Пример:

```
_alignment="LEFT"
_wrap=.T.
? Name PICTURE "!XXXXXXXXXXXXXX" FUNCTION IV15
```

В этом примере поле **Memo** с именем Name разворачивается вертикально в пределах 15 позиций, определенных шаблоном **PICTURE**. Функция вертикальной центрированной развертки перекрывает ранее определенную установку значения переменной **_aligment = "LEFT"**. Если в этом примере задать горизонтальную развертку, то текст будет центрирован между значениями **_lmargin** и **_rmargin**.

Если значение переменной **_wrap = .F.**, то при выводе на печать поля **Memo** с использованием функции **@H** печать начинается с текущей позиции (или определенной значением **_pcolno**) и эта позиция становится левой границей. При этом сохраняются установленные в поле **Memo** границы, абзацы и страницы.

Пример:

```
USE FIO
SET PRINT ON
? NUM "B", OTDEL STYLE "B", FAM STYLE "I"
```

Из базы данных **FIO** на печать выводится жирным шрифтом поля **NUM** и **OTDEL**, а наклонным шрифтом поле **FAM**.

Вместо опций **PICTURE** и **FUNCTION** в команде ? можно использовать функцию **TRANSFORM(<выр.>,<выр.C>)**, которая преобразует заданное выражение в соответствии с указанным в

аргументе **<выр. C>** шаблоном. Эту функцию можно также включать и в другие команды, например **DISPLAY, LABEL, LIST** и **REPORT**.

Пример:

```
USE Kadry
*  Выводит в разрядку И в а н о в
? TRANSFORM(Fam,"@R X X X X X X X X X")
```

Для точного размещения поля **Memo** перед выводом его на экран или на печать с помощью команд ?/?? или **@...SAY...GET** в форме вертикальной развертки нередко целесообразно определить, сколько строк оно занимает. Для этого используется функция **MEMLINES(<имя memo поля>)**, которая возвращает число строк в поле **Memo** при его вертикальной развертке. Для извлечения из поля **Memo** заданной строки используется функция **MLINE(<имя memo поля>,<выр.N>)**.

Выводом на печать обычно управляет драйвер - специальная программа, которая интерпретирует посылаемые на принтер символы. В ряде случаев требуется обращаться к принтеру напрямую, минуя драйвер, например, если драйвер не поддерживает определенные символы. Для этого применяется команда **??? <выр.C>**, которая выводит содержимое заданного выражения непосредственно на принтер, минуя драйвер. Эта команда позволяет передать на принтер символы, не изменяя текущей позиции строки и колонки.

Различные типы принтеров имеют свои собственные управляющие коды, которые приводятся в руководствах по использованию. Управляющие коды могут включать любые печатаемые символы за исключением знака двойных кавычек ("), а также непечатаемые символы, как, например, **Esc**. Непечатаемые символы могут быть определены различными способами, в том числе с помощью спецификаторов. Спецификаторы управляющих символов представляют собой строки, которые идентифицируют непечатаемые символы. Они должные быть заключены в фигурные скобки **{}**.

В табл. 3.8 приводятся спецификаторы управляющих символов.

Предположим, нужно послать управляющий код **Esc-E** (этот код перезагружает принтер) на лазерный принтер Hewlett-Packard. Зная, что код ASCII для Esc - 27, а для E - 69, можно сформировать команду различными способами:

с использованием функции **CHR()**, которая преобразует числовой код в эквивалентный символ ACSII: ??? CHR(27) + "E"

с использованием спецификатора управляющего кода: ??? "{ESC}E"

с использованием только кодов ASCII, заключая их в круглые скобки: ??? "{27}{69}"

с использованием комбинации кодов ASCII и букв: ??? "{27}E"

Таблица 3.8. **Спецификаторы управляющих кодов**

Код ASCII	Спецификатор управляющего кода
0	{ NULL } или { CTRL-@ }
1	{ CTRL-A }
2	{ CTRL-B }
3	{ CTRL-C }
4	{ CTRL-D }
5	{ CTRL-E }
6	{ CTRL-F }
7	{ BELL } или { CTRL-G }
8	{ BACKSPACE } или { CTRL-H }
9	{ TAB } или { CTRL-I }
10	{ LINEFEED } или { CTRL-J }
11	{ CTRL-K }
12	{ CTRL-L }
13	{ RETURN } или { CTRL-M }
14	{ CTRL-N }
15	{ CTRL-O }
16	{ CTRL-P }
17	{ CTRL-Q }
18	{ CTRL-R }
19	{ CTRL-S }
20	{ CTRL-T }
21	{ CTRL-U }
22	{ CTRL-V }
23	{ CTRL-W }
24	{ CTRL-X }
25	{ CTRL-Y }
26	{ CTRL-Z }
27	{ ESC }или{ ESCAPE }или{ CTRL-[}
28	{ CTRL-\ }
29	{ CTRL-] }
30	{ CTRL-^ }
31	{ CTRL-_ }
127	{ DEL } или { DELETE }

Системные переменные представляют собой переменные, которые создаются и поддерживаются автоматически в dBASE IV. Эти переменные обеспечивают форматирование вывода информации на экран и принтер, а также установку параметров принтера. В частности, системные переменные позволяют установить:

характеристики печати, такие как способ подачи бумаги и количество печатаемых копий;

оформление параграфов, например, выравнивание текста, абзац и границы;

общие характеристики печатаемой страницы - шаг и качество печати, длину страницы и сдвиг печати слева.

Имена системных переменных начинаются с символа подчеркивания для отличия от обычных переменных. После запуска dBASE IV автоматически устанавливаются значения системных переменных, принятые по умолчанию. Эти значения

можно изменять при разработке экранных форм, отчетов и марок в УЦ, в командном режиме или в программах. После завершения работы программы с системными переменными, содержащими частные значения, они устанавливаются в первоначальные значения. Команды **CLEAR MEMORY** и **RELEASE** (см. разд. 3.6) не удаляют системные переменные из памяти. Многие системные переменные управляют потоком выходных данных. Этот поток формируется рядом команд, выводящих данные, за исключением команд **@...SAY...GET**, **@...TO** и **EJECT**. Выходной поток обычно направляется на определенное устройство. Устройство, на которое выводится поток выходных данных, можно установить командами **SET CONSOLE**, **SET PRINTER** и **SET ALTERNATE**, а также опцией **TO PRINTER/TO FILE** команды **LIST/DISPLAY**. Выходной поток выводится на экран с текущей позиции курсора, на принтер - с текущей позиции головки принтера и в файл на диске - с текущей позиции указателя записей. Команды **@...SAY...GET** и **@...TO** не создают выходной поток данных, но они могут влиять на последующий вывод выходного потока посредством установки текущей позиции курсора или головки принтера.

Кроме определения устройства для вывода выходного потока, для контроля вывода и других целей, например, сохранения копии выходного потока, предусматривается автоматическое создание альтернативного файла выходного потока, записываемого на диск. Этот текстовый файл, называемый также **ALTERNATE**, записывается параллельно основному выводу командами

SET ALTERNATE TO [<имя файла> [ADDITIVE]]
SET ALTERNATE ON/OFF

Первая форма команды создает и открывает текстовой файл с заданным именем, а вторая (**SET ALTERNATE ON**) записывает в него всю выводимую информацию. Напомним, что полноэкранные команды **@...SAY...GET**, **BROWSE**, **EDIT**, **APPEND** и др. не формируют выходной поток данных, следовательно, альтернативный файл в этом случае не создается.

При записи выходного потока командой ? в начало альтернативного файла записывается пустая строка, так как данная команда всегда предварительно выполняет возврат каретки, что равносильно записи пустой строки. Чтобы избежать этого, следует использовать команду ??. Если файл с заданным именем уже существует, он замещается новым. Команда **SET ALTERNATE TO** без имени файла или команда **CLOSE ALTERNATE** закрывают альтернативный файл.

Опция **ADDITIVE** устанавливает курсор в конец альтернативного файла перед записью в него информации, что позволяет добавлять данные. Команда **SET ALTERNATE OFF** позволяет прервать запись в альтернативный текстовый файл без его закрытия. Затем можно возобновить запись командой **SET**

ALTERNATE ON. Альтернативный текстовый файл не закрывается до тех пор, пока не будет выполнена команда **CLOSE ALTERNATE** или **SET ALTERNATE TO.** Перед открытием этого файла целесообразно проверить, был ли он закрыт или, на всякий случай, выполнить команду **CLOSE ALTERNATE**. Это, впрочем, не лишняя предосторожность и при открытии любого файла. Созданный командой **SET ALTERNATE** файл является стандартным текстовым файлом ASCII и его можно редактировать текстовым редактором dBASE IV, который вызывается командой **MODIFY COMMAND**, или в УЦ выбором позиции **dBASE program** после выбора позиции **<create>** окна генератора прикладных программ. Можно также использовать любой другой текстовый редактор.

Примеры:

```
SET ALTERNATE TO TEXTFILE  && Создает текстовый файл
SET ALTERNATE ON           && Инициирует запись информации
    . . .
SET ALTERNATE OFF          && Приостанавливает запись
    . . .
SET ALTERNATE ON           && Возобновляет запись
    . . .
CLOSE ALTERNATE            && Закрывает текстовый файл
```

Для вывода на экран данных, набираемых с клавиатуры, а также выходного потока данных используется команда **SET CONSOLE ON** (устанавливается по умолчанию). Команда **SET CONSOLE OFF** подавляет вывод на экран выходного потока и набираемых с клавиатуры данных. В частности, не высвечиваются данные, набираемые с клавиатуры в командах **WAIT** или **ACCEPT**, что исключает визуальный контроль введенной информации. При этом не выводятся также сообщения этих команд. Однако не подавляется информация, выводимая командой **@...SAY...GET**, а также сообщения об ошибках и предупреждения при удалении и перезаписи файлов. Эту команду можно использовать только в программах, а не в командном режиме.

Для вывода информации на принтер командами, формирующими выходной поток (например, ?/??, **SET TALK**, **LIST** и других, кроме **@...SAY**), необходимо установить состояние **SET PRINTER ON (OFF** - задается по умолчанию). Вывод на принтер информации командами @...SAY осуществляется командой **SET DEVICE TO PRINTER**.

Команда **SET PRINTER TO** без аргумента восстанавливает заданное по умолчанию устройство вывода (локальное печатающее устройство, общий принтер ЛВС или файл на диске). При работе в среде ЛВС эта команда также выдает сигнал серверу файлов печатать следующее задание. Эта команда имеет еще пять форм:

SET PRINTER TO <устройство DOS>
SET PRINTER TO \\ <имя компьютера>

\<имя принтера> = <назначение>
SET PRINTER TO \\ SPOOLER
SET PRINTER TO \\ CAPTURE
SET PRINTER TO FILE <имя файла>

Первые четыре формы используются для назначения и переназначения вывода информации на различные устройства печати, а пятая создает файлы, форматированные для печати. Первая форма команды выводит информацию на локальный принтер (по умолчанию устройство печати DOS PRN) в один из трех параллельных (LPT1, LPT2, LPT3) или последовательных портов (COM1 или COM2). Эту команду можно также использовать для восстановления установленного по умолчанию устройства печати DOS с одновременным запуском на печать следующего задания в файловом сервере.

Пример:

```
SET PRINTER TO LPT1
```

Формы команды **SET PRINTER TO <аргументы>** со второй по четвертую используются при работе в среде ЛВС. При работе в ЛВС IBM PC и 3Com первый аргумент второй формы задает имя файлового сервера, второй - имя принтера в ЛВС и порт LPT1...LPT3.

Пример:

```
SET PRINTER TO \\SERVER\EPSON=LPT1
```

Пятая форма команды **SET PRINTER TO FILE** применяется для создания форматированного для печати файла и записи его в файл DOS. Если этот файл создается с помощью драйвера печати Ascii.pr2, то формируется текстовой файл ASCII, который можно выводить на любой принтер с построчной печатью или передавать по каналам связи. В dBASE IV по умолчанию создается текстовой файл с расширением **.prt**. Чтобы получить текстовый файл ASCII с расширением **.txt**, нужно использовать драйвер печати Ascii.pr2. Для этого можно присвоить системной переменной **_pdriver** значение "Ascii.pr2". Аналогично выполняются другие назначения драйвера. В частности, для получения файла, форматированного под лазерный принтер Hewlett-Packard LaserJet, следует присвоить: _pdriver = "Hplas100.pr2".

После этого выполняется команда **SET PRINTER TO FILE <имя файла>**. Созданный файл впоследствии можно распечатать на лазерном принтере.

Примеры:

```
COPY A:<имя файла>.PRT COM1
*  Распечатывается файл с диска A: на лазерном принтере
*  подсоединенном к последовательному порту COM1
SET PRINTER TO \\ASERVER\PRINTER=LPT1
```

```
* Назначает вывод информации на устройство печати с именем
* PRINTER и портом LPT1, подключенным к компьютеру с именем
* ASERVER в ЛВС
REPORT FORM Fio TO PRINTER
SET PRINTER TO LPT1 && Выводит информацию из буфера на принтер
```

Изложенный в гл. 2 процесс разработки отчетов и марок в режиме УЦ, управляемом меню, можно реализовать с помощью полноэкранных команд. Команда **CREATE/MODIFY REPORT <имя файла отчета>/?** передает управление на экран разработки (изменения) отчета в УЦ и позволяет создать или изменить файл разработки отчета с расширением **.frm**. Команды **CREATE** и **MODIFY** идентичны. Если указано имя существующего файла, то его можно модифицировать. В противном случае оно рассматривается как имя нового файла.

Опция **?** высвечивает все существующие файлы отчетов, связанные с активным файлом БД, которые можно модифицировать после их выбора. Файл разработки отчета с расширением **.frm** служит для генерации исходной программы выполнения операций по формированию и печати отчета - файла с расширением **.frg**. На основе этого файла, в свою очередь, после обработки псевдокомпилятором создается исполнимый модуль с расширением **.fro**. Если файл с расширением **.frm** уничтожен, то модифицировать существующий отчет с помощью команд **CREATE, MODIFY** или в УЦ уже нельзя. В этом случае исходный программный модуль с расширением **.frg** можно модифицировать с помощью текстового редактора.

При разработке отчета или марки с помощью УЦ или командами **CREATE/MODIFY REPORT (CREATE/MODIFY LABEL)** все заданные для печати параметры можно сохранить на диске в специальном файле формы печати с расширением **.prf**, которому обычно присваивается то же имя, что и отчет или марка. Этот файл автоматически активизируется при очередной модификации или выполнении отчета (марки). Его также можно выбрать с помощью позиции **User print form** меню **Print** генератора отчетов (марок) в УЦ. Кроме того имя файла формы печати можно ввести в системную переменную **_pform** или непосредственно задать значения соответствующих системных переменных, описание которых приводится ниже в данном разделе.

Команда **CREATE/MODIFY LABEL <имя файла отчета>/?** передает управление на экран разработки (изменения) марки в УЦ и позволяет создать (изменить) файл разработки марки с расширением **.lbl**. Эта команда по своему синтаксису и выполнению аналогична предыдущей команде. Из файла разработки марки генерируется исходный программный модуль с расширением **.lbg**, который затем трансформируется псевдокомпилятором в исполнимый объектный модуль с расширением **.lbo**. В dBASE IV можно использовать файлы разработки отчета и марки,

созданные в dBASE III+, которые преобразуются в формат dBASE IV и сохраняются с расширением **.fr3** и **.lb3**.

Как уже указывалось, системные переменные позволяют управлять выходным потоком данных и их печатью. Некоторые из этих переменных уже встречались в тексте и примерах. Рассмотрим подробнее их функции.

В число системных переменных входит группа, которая управляет выходным потоком, и группа управления печатью. Последняя включает подгруппы управления принтером, печатью документа в целом и печатью текста параграфа и пункта (абзаца).

В группу управления выходным потоком входят системные переменные: **_box**, **_pageno**, **_pcolno**, **_pform**, **_plength**, **_plineno**, **_pspacing** и **_tabs**.

Для печати (подавления печати) рамки, заданной командами **DEFINE BOX** или **@...TO**, в зависимости от заданного условия нужно назначить переменной **_box** соответстующее логическое выражение. С помощью этой переменной можно не только напечатать рамку, но и сделать, например, обрамление текста уголками. Для этого нужно прервать печать рамки, установив **_box = .F.**, затем напечатать текст и допечатать оставшуюся часть рамки, восстановив значение **.T.**

Пример:

```
SET TALK OFF
USE Detal ORDER Nomer
DEFINE BOX FROM 5 TO 75 HEIGHT 10 DOUBLE
_box = .T.
?? SPACE(60) AT 10
?
Ind = 1
SCAN
   _box = (Ind < 5 .OR. Ind > (RECCOUNT() - 4))
   ?? Nomer AT 7, Nazvanie
   ?
   Ind = Ind + 1
ENDSCAN
?? SPACE(60) AT 5
?
```

В этой программе удаляется (заполняется пробелами) средняя часть верхней и нижней линии рамки и сама рамка выводится с разрывом по середине, в результате чего выведенные данные из файла БД с именем Detal (номер и название детали) окружаются уголками.

Назначение числового выражения переменной **_pageno = <выр.N>** используется для задания номера страницы первого печатаемого листа при печати комбинированных выходных документов. Кроме того, эту переменную можно использовать

для возвращения текущего номера страницы с целью выполнения на определенной странице заданных действий. В качестве начального значения можно задавать целые числа от 1 до 32767 (по умолчанию 1). Текущий номер страницы определяется увеличением на единицу номера предыдущей страницы и зависит от значений переменных **_plength** и **_plineno**.

Не следует смешивать переменную **_pageno** с **_pbpage**, устанавливающей номер страницы, с которой начинается печать документа. Значения системных переменных **_pageno**, **_pbpage** и **_pepage** должны быть согласованы. При этом, значение переменной **_pageno** должно быть меньше или равно значению переменной **_pepage**. Например, при установке значений **_pbpage = 3**, **_pepage = 5** и **_pageno = 10** печати не будет, так как текущий номер страницы (10) находится вне диапазона печатаемых страниц (с 3-й до 5-й).

Пример:

```
PROCEDURE Zagolovok
PRIVATE _wrap, _alignment
_alignment = "CENTER"
? "-", _pageno PICTURE "@T 999", "-"
?
RETURN .
```

В этом примере расмотрена процедура для печати номера страницы в центре верхней части листа.

Для установки номера колонки (между 0 и 255), с которой начинается вывод данных на принтер, в текущую строку экрана или в файл на диске используется назначение числового выражения переменной **_pcolno = <выр.N>**. Эту переменную можно также использовать для возвращения текущего номера колонки вывода информации, а также для повторной печати по уже напечатанному тексту, если предварительно задать **_wrap = .F.**, **SET PRINTER ON** и установить значение **_pcolno** меньше текущего. Если в этом случае установить **_wrap = .T.**, то текст будет переписываться в буфер принтера без вывода на печать. Следует отметить, что выполняющая аналогичные действия функция **PCOL()** возвращает текущую позицию только головки принтера, а переменная **_pcolno** - текущую позицию выходного потока независимо от установки команды **SET PRINTER**. Действие переменной **_pcolno** эквивалентно опции **AT** команды ?/??.

Пример:

```
_wrap = .F.
? "Пример правки текста"
_pcolno = _pcolno - LEN("правки")
?? REPLICATE("X", LEN("правки"))
?? "Текст изменен"
?
```

Выше уже упоминалось, что установки печати, включая значения системных переменных **_padvance**, **_pageno**, **_pbpage**, **_pcopies**, **_pdriver**, **_pecode**, **_peject**, **_pepage**, **_plength**, **_ploffset**, **_ppitch**, **_pquality**, **_pscode**, **_pspasing** и **_pwait**, можно сохранить при создании отчета в специальном файле с расширением **.prf** (см. раздел 2.6) и впоследствии многократно использовать при печати однотипных отчетов. Активизация этого файла и считывание в оперативную память записанных в нем значений системных переменных выполняется с помощью назначения имени этого файла системной переменной **_pform** = "**<имя файла формы печати>**". Кроме того, эта переменная позволяет возвратить имя активного файла формы печати. Для установки полной длины выводимой страницы от 1 до 32767 строк (по умолчанию 66) используется назначение числового выражения переменной **_plength** = **<выр.N>**. При этом верхний и нижний отступы для заголовка и подножия страницы и размещение текста на листе можно сформировать командой **ON PAGE** (более подробно об этой команде см. ниже в данном разделе).

Для возвращения текущего номера строки выходного потока данных или назначения номера строки, с которой нужно выводить данные, используется переменная **_plineno** = **<выр.N>**. Номер строки увеличивается на единицу после вывода предыдущей строки. Эту переменную можно использовать для позиционирования головки принтера в диапазоне от **0** до **_plength - 1**. В отличие от функции **PROW()** переменная **_plineno** активна и при выключенном принтере (при выводе на экран отслеживаются его строки) и ее значение не может превышать длины страницы. Не следует смешивать данную переменную с функцией **LINENO()**, которая возвращает номер строки выполняемой программы.

Для установки интервала печати между строками при выводе информации можно использовать назначение условных чисел переменной **_pspacing** = **1/2/3**, обозначающих: 1 - одинарный интервал (без пропуска строк - по умолчанию), 2 - двойной интервал (пропуск одной строки), 3 - тройной интервал (пропуск двух строк). Значение этой переменной влияет на высоту печатаемой рамки. Например, двойной интервал вдвое увеличивает заданную высоту.

При вводе и выводе информации часто требуется очередную порцию вводить или выводить с заданной позиции (например, вводить с экрана табличные данные или выводить их на принтер). Для этого на линии разметки экрана в текстовом редакторе устанавливаются символы табуляции, позволяющие передвигать курсор навигационными клавишами по этим позициям. При выводе на принтер позиции табуляции играют роль колонок, с которых начинается печать очередной порции текста. Для установки позиций табуляции можно назначить список этих

позиций в виде символьного выражения переменной **_tabs = <выр.C>**. Позиции в списке отделяются запятыми и располагаются в возрастающем порядке. Например, **_tabs = "5,25,44,65"** устанавливает табуляцию на 5, 25, 44, 65 позициях.

Можно послать символ табуляции, код ASCII которого 9, с помощью команд ?/?? и функции **CHR(9)**, в результате чего головка принтера продвинется до следующего символа табуляции. Если установлено **_wrap = .T.**, то позиции табуляции, равные или большие, чем значение переменной **_rmargin** игнорируются. По умолчанию в текстовом редакторе dBASE IV устанавливается табуляция по 8 позиций. Значение переменной **_tabs** также можно установить как значения параметра **TABS** в файле **Config.db**.

К первой подгруппе группы системных переменных управления печатью относятся: **_padvance, _pdriver, _ploffset, _ppitch, _pquality** и **_pwait**.

К числу параметров управления печатью относится способ продвижения бумаги. Обычно на принтерах предусматриваются два способа продвижения бумаги: построчный и постраничный. В первом выполняется только построчное движение, а во втором - еще и автоматический переход к следующей странице после печати определенного количества строк. Назначение переменной **_padvance = "FORMFEED"/"LINEFEEDS"** ключевых слов позволяет задать способ продвижения бумаги: **FORMFEED** - постраничное продвижение (устанавливается по умолчанию), **LINEFEEDS** - построчное продвижение. В том и другом способе всегда отслеживается количество уже напечатанных строк. При выводе потока выходных данных с построчным продвижением бумаги перед вычислением количества строк, оставшихся до следующей страницы, проверяется, была ли команда на перевод страницы. Если эта команда инициирована, то вычисляется количество строк, которое требуется пропустить по формуле **(_plength - _plineno)**. Данная ситуация возникает в следующих случаях:

инициирована команда **EJECT PAGE** при отсутствии обработчика страницы, устанавливаемого командой **ON PAGE**;

инициирована команда **EJECT PAGE** и командой **ON PAGE** установлен обработчик страницы, но номер текущей строки больше указанного в команде **ON PAGE**;

выполняемая программа печати достигает команд **PRINTJOB** или **ENDPRINTJOB** и системная переменная **_peject** вызывает перевод страницы.

Если перевод страницы выполняется не в режиме потока данных, например, если выдается команда **EJECT** или **SET DEVICE TO PRINTER** и затем выполняется команда **@...SAY...GET**, то для пропуска строк используется формула

(_plength - MOD(PROW(), _plength)).

Посылка на принтер символа **CHR(12)** всегда устанавливает построчный режим продвижения бумаги. Системную переменную

_padvance целесообразно использовать при печати коротких страниц. В этом случае вместо установки длины страницы на принтере можно задать ее в переменной **_plength**, а в переменной **_padvance** - значение **"LINEFEEDS"**.

Пример:

```
PRIVATE _plength, _padvance
_plenth = 30
_padvance = "LINEFEEDS"
USE Sklad
SCAN
   DO Naklad  && Печать накладной
   EJECT      && Перевод страницы в начало следующей накладной
ENDSCAN
CLOSE DATABASE
```

В данном примере печатаются накладные с помощью процедуры Naklad на страницах длиной по 30 строк.

Как уже упоминалось, в принтерах различных типов используются специфические наборы кодов символов. Для преобразования посылаемого из ПК расширенного набора символов IBM, включая управляющие коды конкретного принтера, используется специальная программа - драйвер этого принтера. Например, драйвер интерпретирует посланные в принтер командами (кроме команды ???) символы, обеспечивающие печать различными шрифтами: жирным, с подчеркиванием, наклонным и т.д.

При установке dBASE IV можно назначить требуемый драйвер принтера, который будет использоваться по умолчанию. Его имя автоматически записывается в файл **Config.db**. Если используется другой принтер, то его драйвер можно назначить в программе или в командном режиме с помощью переменной **_pdriver = "<имя файла драйвера принтера>"**. Такое назначение активизирует (открывает) требуемый драйвер принтера, если он имеется на диске. Напомним, что если драйвер находится на диске и в директории, отличном от текущего, то перед именем надо указывать диск и маршрут по директориям. Расширение имени файла драйвера указывать не следует (допускается только расширение **.pr2**). Переменная **_pdriver** позволяет также возвратить имя текущего драйвера принтера. В примере, приведенном ниже, описывается программа для установки требуемого драйвера принтера в зависимости от его типа.

Пример:

```
@ 10,20 SAY "Выберите тип принтера" GET tip
PICTURE "@M Epson FX-85,HP LaserJet, Proprinter";
MESSAGE "Просмотр типов - клавишей ПРОБЕЛ, выбор - клавишей Enter"
READ
DO CASE
   CASE tip = "Epson FX-85"
   _pdriver = "FX85_1.pr2"
```

```
    CASE tip = "HP LaserJet"
    _pdriver = "HPLAS100.pr2"
    CASE tip = "Proprinter"
    _pdriver = "Ibmpro_1.pr2"
OTHERWISE
    _pdriver = "GENERIC.pr2"
ENDCASE
```

Для установки сдвига печати слева от края страницы в диапазоне от 0 до 254 позиций (по умолчанию 0) используется назначение числового выражения переменной **_ploffset** = **<выр.N>**. Значение переменной **_lmargin** отсчитывается от последней позиции, определенной значением **_ploffset**. Обычно эта переменная используется для сдвига печати всего документа, если он ложится не по центру на бумаге. Команда **SET MARGIN** устанавливает то же самое значение, что и переменная **_ploffset**. При этом изменение значения в команде вызывает соответствующее изменение значения переменной.

Значения переменных **_ploffset** и **_lmargin** взаимно независимы. Например, если установлены значения переменных **_ploffset = 10**, **_lmargin = 5**, а также значение переменной, устанавливающей абзац **_indent = 6**, то при изменении значения переменной **_ploffset** остаются неизменными установки левой границы и абзаца относительно сдвига печати, назначенного переменной **_ploffset**.

Установка нескольких типоразмеров шрифта осуществляется посредством назначения ключевых слов переменной **_ppitch** = **"PICA"/ "ELITE"/"CONDENSED"/"DEFAULT"** (**PICA** - 10 символов на дюйм, **ELITE** - 12 символов на дюйм, **CONDENSED** - примерно 17.16 символов на дюйм, **DEFAULT** - установленный на принтере). Кроме того, эта переменная, как и другие, позволяет определить (возвратить) текущее значение типоразмера шрифта.

Принтеры обычно имеют режимы качественной и черновой печати текста. Для установки того или иного режима используется назначение логического выражения переменной **_pquality** = **<логическое выражение>**. При значении логического выражения **.T.** устанавливается режим качественной печати текста, а при значении **.F.** - черновой печати. При одновременной установке режима печати на принтере и с помощью переменной преимущество имеет назначение на принтере.

Для установки режима ожидания перед печатью страницы используется назначение логического выражения переменной **_pwait** = **<логическое выражение>**. При значении логического выражения **.T.** устанавливается режим ожидания перед печатью страницы, а при значении **.F.** печать выполняется непрерывно. Если для вывода на печать используется команда **PRINTJOB**, то назначение переменной **_pwait** нужно делать перед этой командой.

Ко второй подгруппе группы управления печатью относятся системные переменные: **_pbpage**, **_pcopies**, **_pecode**, **_peject**, **_pepage** и **_pscode**. Системные переменные этой подгруппы активизируются командой **PRINTJOB**.

При печати выходных документов нередко требуется перепечатать одну или несколько страниц или продолжить печать отчета, прерванную из-за неполадок принтера или по другим причинам. Для задания первой печатаемой страницы (в диапазоне от 1 до 32767, по умолчанию устанавливается 1) используется назначение числового выражения переменной **_pbpage** = **<выр.N>**. Значение этой переменной должно быть меньше или равно значению переменной **_pepage**, которая устанавливает последнюю печатаемую страницу документа (также в диапазоне от 1 до 32767, по умолчанию устанавливается 32767) посредством назначения ей числового выражения: **_pepage** = **<выр.N>**. При этом текущий номер страницы (значение переменной **_pageno**) должен быть больше или равен номеру первой печатаемой страницы и меньше или равен номеру последней печатаемой страницы.

Для того чтобы перепечатать заданное количество страниц, назначают совместно значения переменным **_pbpage** и **_pepage**. Следует отметить, что номера страниц в переменных **_pbpage**, **_pepage** и **_pageno** обозначают не их физические номера, а номера, печатаемые на странице. При этом первой физической странице в переменной **_pageno** может быть присвоен любой номер, от которого начинается отсчет. При перепечатке страниц из пронумерованного текста нужно также задавать не физические номера перепечатываемых страниц, а их порядковые номера, соответствующие принятой нумерации страниц.

Пример:

```
sled = 1
@ 10,30 SAY "Печатать со страницы" GET sled PICTURE "999"
READ
_pbpage = sled
DO Otchet
```

В этом фрагменте программы осуществляется запуск на продолжение печати прерванного отчета со страницы, номер которой вводится с клавиатуры.

Установка числа печатаемых копий (от 1 до 32767, по умолчанию 1) осуществляется назначением числового выражения переменной **_pcopies** = **<выр.N>**. Эту переменную можно использовать только в программах вместе с командами **PRINTJOB** и **ENDPRINTJOB**, установив предварительно ее значение перед выполнением этих команд.

Ниже приводится фрагмент программы для задания количества печатаемых копий.

Пример:

```
ncop = 1
@ 10,20 SAY "Число копий: " GET ncop PICTURE "99"
_pcopies = ncop
READ
_pscode = "{27}{38}{108}{49}{79}"
PRINTJOB
      .       && Команды печати
      .
      .
   _pecode = "{27}{69}"
ENDPRINTJOB
```

Для формирования стиля печати на принтер направляются управляющие коды. Эти коды посылаются командами ?/??, ??? или посредством назначения переменным **_pscode** и **_pecode**. Обычно команды ?/?? используются для формирования стиля печати отдельных строк текста, команда ??? - для формирования стиля печати параграфов, а значения системных переменных **_pscode** и **_pecode** - для всего печатаемого материала. Поэтому их назначение часто выполняется в командном режиме перед началом печати. В программе значение переменной **_pscode** назначается до выполнения команды **PRINTJOB**, а значение переменной **_pecode** - до выполнения команды **ENDPRINTJOB**. Управляющие коды, назначенные переменной **_pscode**, посылаются на принтер командой **PRINTJOB**, а назначенные переменной **_pecode** - командой **ENDPRINTJOB**.

Переменным назначаются коды в виде символьных выражений: **_pscode = <выр.C> и _pecode = <выр.C>**. В качестве кодов разрешаются числа от 0 до 255, а длина символьной строки не должна превышать 255 позиций.

Установка перевода бумаги осуществляется посредством назначения ключевых слов переменной **_peject = "BEFORE"/"AFTER"/"BOTH"/"NONE"**, где **BEFORE** (по умолчанию) переводит лист перед печатью; **AFTER** - после печати; **BOTH** - до и после печати; **NONE** - не переводит лист. Эта переменная используется только в программах и помещается перед командой **PRINTJOB**.

Не следует смешивать действие этой переменной с командой **EJECT**, которая вызывает перевод листа на принтере в соответствии с режимами продвижения бумаги, установленными в переменной **_padvance**.

К третьей подгруппе управления печатью относятся системные переменные: **_alignment**, **_indent**, **_lmargin**, **_rmargin** и **_wrap**. Эти переменные позволяют задать позиционирование печатаемого текста, величину отступа абзаца и границы печати текста. Прежде чем позиционировать текст, надо задать его границы с помощью переменных **_lmargin** и **_rmargin**. Назначение числового выражения переменной **_lmargin = <выр.N>**

определяет на странице левую границу вывода информации командой ?, если **_wrap = .T.** в диапазоне от 0 до 254.

Напомним, что действия переменных **_lmargin** и **_ploffset** различны. Значение последней переменной определяет расстояние между левым краем бумаги до позиции, с которой начинается печать (сдвиг печати), если **_lmargin = 0**. Значение **_lmargin** определяет позицию печати относительно сдвига, задаваемого значением **_ploffset**.

Пример:

```
SET TALK OFF
USE Detal ORDER Nomer
_wrap = .T.
_imargin = 0
? "Список деталей"
?
_lmargin = 15
SCAN
  ? Nomer + "   ", Nazvanie
ENDSCAN
```

Назначение числового выражения переменной **_rmargin = <выр.N>** устанавливает правую границу вывода информации командами ?/??, если **_wrap = .T.** Правая граница представляет собой первую колонку после печати текста на строке. Минимальное значение этой переменной должно быть наибольшим из **(_lmargin + 1)** и **(_lmargin + _indent + 1)**, где значение **_indent** может быть отрицательным числом. Максимальное значение переменной 255. После установки значений переменных **_lmargin** и **_rmargin** можно форматировать выводимый текст внутри заданных границ.

Переменная **_alignment** позволяет установить способ позиционирования текста, выводимого командами ?/??, по отношению к границам при условии, что **_wrap = .T.**, посредством назначения ключевых слов: **_alignment = "LEFT"/"CENTER"/"RIGHT"**, где **LEFT** выравнивает текст по левой границе; **CENTER** центрирует текст между левой и правой границами; **RIGHT** выравнивает текст по правой границе. По умолчанию действует **LEFT**. Не следует смешивать применение этой системной переменной с функциями поля **@B**, **@I** и **@J**, действующими аналогично, но по отношению к тексту в поле.

Пример:

```
_wrap = .T.
SET PRINTER ON
SET TALK OFF
_alignment = "CENTER"
?? "-", STR(_pageno,3,0),"-"
SET PRINTER OFF
_alingnment = "LEFT"
```

Переменная **_indent** позволяет установить величину отступа первой строки каждого нового параграфа, печатаемого командой ?. Количество позиций отступа абзаца устанавливается назначением переменной числового выражения: **_indent = <выр.N>** в диапазоне от отрицательного значения - **(_lmargin)** до **(_rmargin - _lmargin - 1)**. Сумма **(_indent + _lmargin)** должна быть меньше значения **_rmargin**.

Выше при описании многих системных переменных упоминалась переменная **_wrap**, которая устанавливает или отменяет в зависимости от своего значения вертикальную развертку текста командами ?/?? внутри заданных переменными **_lmargin** и **_rmargin** границ. Значением этой переменной должно быть логическое выражение **_wrap = <логическое выражение>**. Если логическое значение **.T.**, то развертка выполняется, в противном случае - нет. Системные переменные **_alignment**, **_indent**, **_lmargin** и **_rmargin** действуют только при **_wrap = .T.**

При выводе на печать отчетов часто требуется на страницу выводить верхний и нижний колонтитулы. Это можно сделать с помощью команды (состояния)

ON PAGE [AT LINE <выр.N> <команда>

которая позволяет переключить процесс на выполнение действий, указанных в ней (обычно вызовом процедуры) при переходе на строку, заданную в опции **AT LINE** в ходе выполнения команд ?/?? или **EJECT PAGE**. Команду **ON PAGE** иногда называют обработчиком страницы. Текущий номер строки содержится в системной переменной **_plineno**. Если для создания отчета используется генератор отчетов, то команда **REPORT FORM** при печати отчета автоматически активизирует обработчик страниц. Обработчик страниц может использоваться и с другими командами вывода на печать, такими как **DISPLAY** или **SCANE**. При определении номера строки следует учитывать, что в длину страницы входят высота верхнего и нижнего колонтитулов страницы.

Процедура печати верхнего и нижнего колонтитулов должна начинаться командой ?, чтобы подавить перевод строки при активизации команды **ON PAGE**, а заканчиваться командой **EJECT PAGE**.

Все процедуры, которые обеспечивают вывод различных частей отчета (введения, итогов, подытогов и т.д.), должны начинаться командой ? и заканчиваться командой ??. В качестве команд, выполняемых в состоянии **ON PAGE**, запрещено использовать те же команды, что и при создании функции пользователя (см. разд. 3.6).

Пример:

```
ON PAGE AT LINE 60 DO Page_brk
SET TALK OFF
```

```
SET PRINTER ON
DO Header    && Печать верхнего колонтитула первой страницы
SCAN ALL
  ? Otd,Fam,Name
ENDSCANE
EJECT PAGE   && Активизизация обработчика страницы
DO Footer    && Печать нижн. колонтитула послед. стр.
SET PRINTER OFF
ON PAGE      && Деактивизация обработчика страницы
SET TALK ON
RETURN
PROCEDURE Header
EJECT PAGE   && Переход на новую страницу
?            && Переход на вторую строку
? "Список сотрудников" STYLE "B", DATE() AT 70
?
?            && Печать списка сотрудников с седьмой строки
RETURN
PROCEDURE Footer
?
?            && Печать нижнего колонтитула на 63 строке
? "Учетный номер " Uchnom
?? "Стр." AT 70, LTRIM(STR(_pageno,4,0))
RETURN
PROCEDURE Page_brk
DO Footer
DO Header
RETURN
```

Для обеспечения перевода бумаги на принтере к началу следующей страницы используется команда **EJECT**. В зависимости от установки в системной переменной **_padvance** значения **"FORMFEEDS"** или **"LINEFEEDS"** на принтер посылается код ASCII 12 или 10. После выполнения команды **EJECT** значения функций **PROW()** и **PCOL()** устанавливаются в нуль. Перед выполнением команды **EJECT** для проверки состояния принтера можно использовать функцию **PRINTSTATUS()**, которая возвращает логическое значение .T., если принтер включен. Эта функция возвращает состояние последнего установленного печатающего устройства независимо от способа установки, например командами **SET PRINTER ON**, **SET DEVICE TO PRINTER** или любой командой, содержащей предложение **TO PRINTER**, а также клавишами **Ctrl-P**. Следует отметить, что команда **EJECT** не изменяет значения системных переменных **_pageno** и **_plineno**.

Пример:

```
DO WHILE .T.
@ 23,25 SAY "Включите принтер"
@ 24,25 SAY "Для запуска отчета на печать нажмите клавишу W" ;
        GET Gotov "!"
```

```
READ
IF .NOT. PRINTSTATUS() .AND. Gotov = "W"
   LOOP
ENDIF
IF Gotov = "W"
   REPORT FORM Kadry TO PRINT
ENDIF
EXIT
ENDDO
```

Более широкими возможностями обладает команда **EJECT PAGE**, которая кроме перевода бумаги в начало страницы может вызывать обработчик страницы, если установлена команда **ON PAGE**. В последнем случае команда **EJECT PAGE** сравнивает текущую строку печати со строкой, указанной в команде **ON PAGE**.

Если текущий номер строки меньше этой величины, то команда **EJECT PAGE** переводит головку принтера на соответствующее число строк, чтобы вызвать обработчик страницы. Если же текущий номер строки больше указанного в команде **ON PAGE** или обработчик не установлен, то выполняются следующие действия. При установке команды **SET PRINTER ON** и **_padvance = "FORMFEED"** осуществляется перевод страницы. В случае **_padvance = "LINEFEEDS"** на принтере переводится число строк, необходимое для перевода страницы.

При выводе на другое устройство, например, на экран или в файл на диск (при установке **SET CONSOLE** или **SET ALTERNATE**), выполняется пропуск такого же числа строк. После выполнения команды **EJECT PAGE** номер страницы в системной переменной **_pageno** увеличивается на единицу, а номер строки в **_plineno** устанавливается в нуль.

Выше указывалось, что для вывода данных на печать основной является команда ?/??. Однако перед печатью часто требуется выполнить подготовительные действия, например, перевести страницу, установить в начальное положение головку принтера и т.д. Для выполнения этих и других вспомогательных действий служит команда

PRINTJOB
<команды>
ENDPRINTJOB

Эта команда состоит из двух частей. Первая из них **PRINTJOB** активизирует параметры печати, которые определены системными переменными, и осуществляет следующие действия:

посылает начальные коды печати, определенные системной переменной **_pscodes**;

переводит страницу, если системная переменная **_peject** имеет значение **"BEFORE"** или **"BOTH"**;

устанавливает **_pcolno** в 0.

Вторая часть команды **ENDPRINTJOB** вызывает следующие действия:

посылает конечные коды печати, определенные системной переменной **_pecodes**;

переводит страницу, если переменная **_peject** установлена в **"AFTER"** или **"BOTH"**;

возвращает управление команде **PRINTJOB** столько раз, сколько это определено переменной **_pcopies**.

Системные переменные обычно устанавливаются перед командой **PRINTJOB**. При необходимости эти переменные переопределяются после команды **ENDPRINTJOB**.

Кроме указанных выше можно использовать все другие системные переменные, например, для остановки печати перед каждой страницей или для назначения формата печати. При разработке программ следует учесть, что команды **PRINTJOB** и **ENDPRINTJOB** не могут быть вложены сами в себя.

Пример 1:

```
USE Fio
_peject = "AFTER"
SET PRINT ON
PRINTJOB
  ON PAGE AT LINE _plength-1 DO Pagebrk
  SCAN
    ?? Otd AT 0,Fam AT 8,Name AT 23
    ?
  ENDSCAN
ENDPRINTJOB
SET PRINT OFF
RETURN
PROCEDURE Pagebrk
EJECT PAGE
?? DATE() AT 1,"Стр." AT 67,_pageno PICTURE "999" AT 73
?
RETURN
```

Пример 2:

```
_pbpage = 10 && Начальная страница печати
_peject  = "AFTER" && Перевод страницы после печати
_pcopies = 3 && Число копий
REPORT FORM Otchet TO PRINT
_pbpage = 1  && Начальная страница по умолчанию
_pcopies = 1 && Число копий по умолчанию
```

В последнем примере печатаются три копии с десятой страницы и до конца отчета и переводится страница после печати. Для печати отчета используется команда

REPORT FORM <имя файла отчета>/?
[<границы>] [WHILE <условие>]

[FOR <условие>] [PLAIN]
[HEADING <выр.C>] [NOEJECT]
[TO PRINTER/TO FILE <имя файла>] [SUMMARY]

которая запускает программу печати отчета (файл отчета), созданную командой **CREATE/MODIFY REPORT**.

Эта программа (здесь не показана) включает команды **PRINTJOB** и **ENDPRINTJOB**. Программа печати выводит все или часть записей данных из активного файла БД или выборки, определенных заданными условиями отбора. Если в программе печати определены группы, активный файл БД должен быть проиндексирован или отсортирован по тем же полям, которые входят в выражение, задающее в отчете группу (см. разд. 2.6). Для одного файла БД может быть разработано несколько программ печати отчетов различного вида. Вызов списка всех программ печати, относящихся к активному файлу БД, выполняется командой **REPORT FORM ?**. Отчет можно вывести на экран, принтер или в текстовый файл на диск. Имя файла отчета (программы печати отчета), указанного в опции **FORM**, можно задавать с расширением **.frg** или **.fro**, с другим расширением, а также без расширения. В последних двух случаях выполняется поиск файла разработки отчета с заданным именем и, если он найден, то на его основе генерируется исходный программный файл с расширением **.frg**, который перед выполнением компилируется в объектный модуль с расширением **.fro**. В данной команде можно также использовать файл разработки отчета, созданный в dBASE III+. Этот файл преобразуется в формат dBASE IV, далее выполняются описанные выше операции по его преобразованию, а в исходном файле dBASE III+ расширение меняется на **.fr3**.

При работе в среде ЛВС команда **REPORT FORM** осуществляет автоматический захват файла БД перед печатью. Если другой пользователь пытается захватить файл, высвечивается сообщение **File is in use by another** (Файл захвачен другим пользователем).

Опция **PLAIN** при печати подавляет вывод нижнего и верхнего колонтитула на всех страницах, кроме первой. Опция **HEADING** выводит дополнительный заголовок на первой строке каждой страницы. Опции **PLAIN** и **HEADING** - взаимно исключающие, причем **PLAIN** имеет приоритет.

Опция **NOEJECT** с ключевым словом **TO PRINTER** подавляет перевод страницы перед печатью отчета и начинает печатать его на текущей странице. Если при создании отчета использовалась опция развертки текста по знаку точка с запятой (;) в символьных или полях **Memo** (состояние **Wrap semicolons ON**), то этот знак в тексте не печатается, а вызывает возврат каретки и перевод строки.

Опция **TO FILE** - выводит отчет в текстовый файл (**.txt**). на устройство, установленное по умолчанию. Опция **SUMMARY**

подавляет вывод на экран всех строк отчета, кроме итогов и подытогов.

Для печати марок используется аналогичная по структуре и выполнению команда

**LABEL FORM <имя файла марок>/? [<границы>] [SAMPLE]
[WHILE <условие>] [FOR <условие>]
[TO PRINTER/TO FILE <имя файла>]**

которая выводит на принтер или в файл на диск марку, созданную ранее командой **CREATE/MODIFY LABEL**. Здесь отметим лишь, что имя файла марок можно задавать с расширениями **.lbg** и **.lbo**. Опция **SAMPLE** позволяет выполнить пробную печать одного ряда марок для контроля их расположения на листе бумаги. Вместо значений полей печатаются буквы **X**. Можно распечатать несколько таких образцов, отвечая **Y** на предложение **Do you want more samples?** (Печатать еще образцы?), высвечиваемое в строке сообщений. При ответе **N** будут печататься марки с данными из файла БД.

Текстовой файл отчета (или любой другой текстовой файл), записанный на диск, можно распечатать командой

**TYPE <имя файла> [TO PRINTER/
TO FILE <имя файла>] [NUMBER]**

Эта команда (без опций) выводит на экран текстовый файл, включающий только коды ASCII. Коды расширенного набора символов выводятся в виде графических символов. Имя файла должно включать расширение.

Следует отметить, что данная команда не пригодна для просмотра файлов БД, так как эти файлы содержат символы, которые используются в dBASE IV для управления. Команда **TYPE** не выводит открытые файлы, например программа не может включать команду **TYPE** для вывода самое себя.

Опция **TO PRINTER** выводит текстовый файл на печать. Если драйвер принтера не поддерживает графические символы, то вместо них используются обычные, например, пробел, знак плюс и т.д. Ключевое слово **NUMBER** дает возможность высвечивать или печатать номер строки. Это часто требуется при выводе программ. Номер строки соответствует строкам команд, а не высвечиваемым на экране строкам, так как команды могут продолжаться на нескольких строках. Если установлена команда **SET HEADING ON**, то в начале каждой страницы высвечивается имя файла, текущая дата в формате, установленном командой **SET DATE** и номер страницы.

Команда **SET ODOMETER TO <выр.N>** изменяет интервал вывода номеров строк с 1 (задан по умолчанию) до 200. Эта команда также определяет интервал изменения значений счетчика записей во всех командах, которые выводят номера записей.

Команда **SET TALK OFF** подавляет вывод номера записи на экран. Опция **TO FILE** позволяет записывать текстовой файл на диск, что дает возможность скопировать выводимый файл с другим именем.

Описанные выше команды не обеспечивают вывода на печать графических символов. Исключение составляет команда

DEFINE BOX FROM <колонка> TO <колонка>
HEIGHT <выр.N> [AT LINE <номер строки>]
[SINGLE/DOUBLE/<символы определения рамки>]

с помощью которой печатается рамка вокруг текста. Для этого нужно определить позицию левой (после слова **FROM**) и правой (после слова **TO**) сторон рамки, строку верха рамки и ее высоту. Если координаты рамки меньше текущих координат принтера (или отсутствует опция **AT LINE**), то рамка печатается на текущей строке принтера.

Символы определения рамки указываются аналогично команде **SET BORDER** (см. разд. 3.2). По умолчанию печатается одинарная рамка. Опция **HEIGHT** определяет высоту рамки, а опция **SINGLE (DOUBLE)** - одинарную (двойную) линию рамки. Для печати рамки нужно установить значение **.T.** системной переменной **_box**.

Кроме команд собственно вывода на печать в командном языке dBASE IV предусмотрены команды установки (**SET**-команды) режимов вывода на печать информации, параметров печати и т.д. К ним относится команда **SET CURRENCY TO [<выр.C>]**, устанавливающая строку до 9 символов для обозначения валюты, и команда **SET CURRENCY LEFT/RIGHT**, определяющая размещение слева или справа от числа строки символов валюты.

Для назначения вывода заданного количества десятичных знаков чисел после запятой в диапазоне от 0 до 18 (по умолчанию установлено два знака) используется команда **SET DECIMALS TO <выр.N>**.

С помощью команды **SET SEPARATOR TO [<выр.C>]** можно задать отличающийся от принятого по умолчанию (запятая) разделитель трех десятичных порядков в целой части числа.

Пример:

```
SET SEPARATOR TO "."
•SET POINT TO ","
Val = "100000.55"
? Val PICTURE "999,999.99" && Выводит 100.000,55
```

В различных странах принят свой формат записи даты. Для установки различных форматов даты используется команда

SET DATE [TO] AMERICAN/ANSI/BRITISH/FRENCH/GERMAN/ ITALIAN/JAPAN/USA/MDY/DMY/YMD

описание ключевых слов которой, приводится ниже.

AMERICAN	ММ/ДД/ГГ
ANSI	ГГ.ММ.ДД
BRITISH/FRENCH	ДД/ММ/ГГ
GERMAN	ДД.ММ.ГГ
ITALIAN	ДД-ММ-ГГ
JAPAN	ГГ/ММ/ДД
USA	ММ-ДД-ГГ
MDY	ММ/ДД/ГГ
DMY	ДД/ММ/ГГ
YMD	ГГ/ММ/ДД

Пример:

```
Date = CTOD ("30/07/89")
SET DATE ANSI
? Date         && Высвечивает 89.30.07
SET DATE GERMAN
? Date         && Высвечивает 30.07.89
```

Разделитель в дате можно изменить командой **SET MARK [<выр.С>]** (по умолчанию установлен раделитель /).

Пример:

```
SET MARK TO "."
? DATE()       && Высвечивает 10.08.89
```

Обычно при выводе даты год представляется двумя последними цифрами. По умолчанию устанавливается состояние **SET CENTURY OFF** и числом века предполагается 19, которое в этом случае нельзя ни вводить, ни выводить в дате. Однако в вычислениях допускается получение даты за пределами двадцатого века, которая будет храниться в формате внутреннего представления даты. Если такая дата редактируется с помощью полноэкранных команд в состоянии **SET CENTURY OFF**, то число века изменяется на 19. Команда **SET CENTURY ON** позволяет при вводе и выводе даты число века представлять четырьмя цифрами.

Пример:

```
SET DATE GERMAN
? DATE()       && Высвечивается, например, 30.07.89
SET CENTURY ON
? DATE()       && Высвечивается 30.07.1989
SET CENTURY OFF
```

В приведенных выше примерах использовались функции **CTOD()** для преобразования даты, представленной в виде символьного выражения в переменную типа даты, и **DATE()** для возвращения значения даты (более подробно о функциях обработки дат см. разд. 3.7).

Для установки левой границы начала печати (сдвиг печати) всего текста используется команда

SET MARGIN TO <выр.N>

которая задает размеры левого поля (по умолчанию равно нулю). Это поле также устанавливается системной переменной **_ploffset**. Действующим будет последнее установленное командой или переменной значение поля. Установка левого поля может быть также выполнена посредством присвоения значения соответствующему параметру в файле **Config.db**. Значение системной переменной **_lmargin** добавляется к значению, установленному командой **SET MARGIN**, если значение системной переменной **_wrap** равно **.T.**

При выводе информации командами ?/?? поля данных автоматически разделяются пробелом. В некоторых случаях требуется сцеплять выводимые данные из нескольких полей. Это можно сделать различными способами, в частности установкой состояния **SET SPACE OFF** (**ON** задается по умолчанию).

Пример:

```
Fam = 'Сергеев'
Nam = 'Юрий'
? Fam,Nam    && Выводит: Сергеев Юрий
SET SPACE OFF
? Fam,Nam    && Выводит: СергеевЮрий
```

3.6. РАЗРАБОТКА ПРИКЛАДНЫХ ПРОГРАММ

В предыдущих разделах были описаны команды создания файлов БД, ввода и обновления в них данных, поиска и отбора данных, создания экранных форм ввода и вывода информации. После освоения этого материала можно приступить собственно к разработке прикладных программ.

Прикладная программа представляет собой последовательность команд, постоянно хранящихся в файле на диске с расширением **.prg** для исходных программ на языке dBASE IV и с расширением **.prs** для программ на языке **SQL**. Напомним, что непосредственно выполняются исполнимые программные модули, которые формируются посредством псевдокомпиляции исходных программ. Для создания исходных программных файлов используются обычные текстовые редакторы, например встроенный редактор dBASE IV, который вызывается командой

MODIFY COMMAND/FILE <имя файла>
[WINDOW <имя окна>]

Этот редактор используется для создания и модификации программных файлов и стандартных текстовых файлов ASCII.

Кроме того, он автоматически вызывается при разработке экранных форм, отчетов и марок. Если имя файла не задано или не найдено, то открывается экран для создания нового файла. Напомним общее правило при вызове файла. Если не указан диск и маршрут по директориям вызываемого файла, то предполагается текущий диск и директорий, а если не указано расширение файла, то для данной команды предполагается расширение **.prg**. При использовании альтернативного синтаксиса команды **MODIFY FILE** расширение по умолчанию не действует и его надо задавать (если, конечно файл, создавался с расширением). По умолчанию в текстовом редакторе максимальная длина строки составляет 1024 символа (позиции) и всего можно создать 32000 строк. Опция **WINDOW** позволяет вводить текстовой файл в заданное и ранее созданное окно. Если такого окна нет, то высвечивается сообщение **Window name not found** (Окно с заданным именем не найдено). Вместо встроенного редактора можно использовать любой текстовый редактор, установив его имя в файле **Config.db**. После изменения существующего файла предыдущая версия автоматически сохраняется на диске с тем же именем и расширением **.bak**. Для обеспечения функций по созданию и модификации файлов в текстовом редакторе dBASE IV имеется меню, которое описано в разд. 2.5 и 2.6.

В программе команда записывается на одной или нескольких строках. Конец командной строки отмечается клавишей **Enter**. При записи команды в несколько строк каждая из них, кроме последней, должна заканчиваться точкой с запятой. Созданный программный файл вызывается в оперативную память и затем выполняется командой **DO** (более подробно об этой команде см. ниже в данном разделе).

Как уже упоминалось, после загрузки в оперативную память исходного программного файла псевдокомпилятор dBASE IV автоматически компилирует его в объектный исполняемый модуль, который затем выполняется. Этот модуль записывается на диск с расширением **.dbo** для обеспечения повторных запусков. Можно также откомпилировать исходную программу командой

COMPILE <имя файла> [RUNTIME]

которая считывает файл с исходной программой и создает выполнимый объектный модуль.

Опция **RUNTIME** выводит на печать команды, которые не разрешается использовать при выполнении программы с помощью модуля **RUNTIME**, включающего в себя только интерпретатор и не содержащего интерфейса пользователя. Данная команда позволяет компилировать не только исходную программу с расширение **.prg**, но и программы на языке **SQL**, программные файлы экранных форм с расширением **.fmt**,

отчетов (**.frg**), марок (**.lbg**), запросов на отбор (**.qbe**) и на обновление данных (**.upd**). После компиляции все эти файлы записываются с расширением **.dbo**.

Прикладная программа обычно состоит из главной процедуры и вызываемых из нее процедур, например:

```
<команды>          && Начало главной процедуры
DO Proc1
DO Proc2
DO Proc3
RETURN             && Конец главной процедуры
PROCEDURE Proc1    && Начало Proc1
<команды>
RETURN             && Конец Proc1
PROCEDURE Proc2    && Начало Proc2
<команды>
RETURN             && Конец  Proc2
PROCEDURE Proc3    && Начало Proc3
<команды>
RETURN             && Конец  Proc3
```

Процедура представляет собой программный модуль, который выполняет определенную задачу. После ее выполнения управление передается в вызвавшую или главную процедуру. Процедура так же, как и программа, вызывается и выполняется командой

DO <имя программы>/<имя процедуры>
[WITH <список параметров>]

Опция **WITH** передает в вызываемую процедуру параметры (до 64), которые представляют собой выражения, разрешенные в dBASE IV. Если в выражении используется имя переменной, одинаковое с именем поля, то предполагается, что передается поле (поля имеют приоритет перед переменными). Для того чтобы передавалась переменная, нужно перед ее именем поставить **M->**.

В любой процедуре можно вызывать другие процедуры, формируя, таким образом, вложенные структуры процедур. Следует избегать рекурсивного вызова процедур, т.е. вызова из процедуры этой же процедуры. Процедуры можно помещать вместе с главной процедурой или отдельно в специальный файл процедур, в котором может размещаться до 1170 процедур. Практическим ограничением на число процедур обычно выступает емкость оперативной памяти, в которой размещается для выполнения как программный файл, так и файл процедур. В начале каждого объектного модуля помещается список процедур, которые в нем используются независимо от того, находятся ли они вместе с главной процедурой программы или в процедурном файле. В этот список первым входит и сама главная процедура под именем программного файла. Если

программный файл не имеет имени, то по умолчанию в качестве его процедурного имени назначается **Main**.

Размещение процедур в программном файле обеспечивает более быстрое их выполнение, так как не требуется открывать и закрывать процедурный файл и считывать процедуры в оперативную память для выполнения. В то же время процедуры, которые выполняются только один раз, занимают оперативную память в течение всего времени выполнения прикладной программы. Поэтому целесообразно размещать часто выполняемые процедуры вместе с главной процедурой, а редко вызываемые - в процедурном файле. Перед выполнением команды **DO** вызова процедуры осуществляется поиск процедуры с заданным именем в следующей последовательности: в процедурном файле, если он открыт, отыскивается открытый объектный модуль с именем искомой процедуры; отыскивается и открывается закрытый объектный модуль с именем искомой процедуры; отыскивается и компилируется исходный программный модуль с именем искомой процедуры; отыскивается и компилируется программа **SQL** с расширением **.pfs** и именем искомой процедуры. Как видно из этой схемы поиска, в качестве процедуры можно также рассматривать и обычные программные модули, хранящиеся на диске. Это позволяет разработать гибкую схему вызова процедур с целью обеспечения оптимального использования оперативной памяти. Каждая процедура ограничивается объемом 64 Кбайт. Одновременно может быть открыто до 32 процедур, в число которых входят объектные модули разработанных ранее отчетов, марок, экранных форм и запросов.

Если установлена команда **SET DEVELOPMENT ON**, то перед выполнением команды **DO** сравнивается время и дата создания исходного программного и объектного модулей с одним и тем же именем. Если исходный модуль создан позже, то он автоматически компилируется псевдокомпилятором и создается новый объектный модуль. После редактирования исходного модуля с помощью встроенного текстового редактора dBASE IV, вызываемого командой **MODIFY COMMAND** или из УЦ, выполняется автоматическая компиляция перед его сохранением и новый объектный модуль замещает старый.

Каждая процедура, размещаемая в файле процедур или в программном файле, должна начинаться с команды

PROCEDURE <имя процедуры> и завершаться командой

RETURN [<выражение>/TO MASTER/TO <имя процедуры>]

Любая программа, а также и функция, определенная пользователем, должны завершаться командой **RETURN**, которая передает управление вызывающей программе, в УЦ или в командный режим. Если эта команда пропущена, то она

автоматически помещается в конце программы, процедуры или функции. При возврате управления процедурный файл закрывается (если процедура вызывалась из него) и выполняется команда в программе, следующая за вызвавшей процедуру командой. Опция **TO MASTER** выполняет возврат в программу высшего уровня (в DOS или в dBASE IV) в зависимости от того, откуда был осуществлен вызов. Опция **TO <имя процедуры>** возвращает управление в активную процедуру с указанным именем. Команда **RETURN** удаляет все переменные, которые были определены в процедуре как **PRIVATE** (переменные, определенные как **PUBLIC** остаются без изменения, об определении переменных см. ниже), а также очищает значение (номер последней ошибки), возвращаемое функцией **ERROR()**.

Для того чтобы обеспечить выход из программного файла в DOS, используется команда **QUIT**, которая закрывает все открытые файлы, завершает сеанс работы dBASE IV и возвращает управление операционной системе. Данная команда обеспечивает единственный надежный способ выхода из dBASE IV. Выход из dBASE IV путем перезагрузки ПК может повредить открытые файлы и вызвать потерю данных.

Если установлено состояние **SET ESCAPE ON** (задается по умолчанию), то выполнение программы можно приостановить нажатием клавиши **Esc**, а поток выводимой на экран информации клавишами **Ctrl-S**. Состояние **SET ESCAPE ON** также разрешает занесение кода этих клавиш в буфер клавиатуры **HISTORY**. После приостановки программы высвечивается сообщение

*****INTERRUPTED*****

Cancel, Ignore, Suspend? (C,I or S)

позволяющее прекратить (**Cancel**), продолжить (**Ignore**), приостановить (**Suspend**) выполнение программы. После приостановки управление передается в командный режим, откуда можно возобновить выполнение программы командой **RESUME** со следующей командной строки, в которой произошла приостановка. При установке состояния **SET ESCAPE OFF** аварийное прекращение выполнения программы возможно только перезапуском ПК. Команда **SUSPEND** позволяет приостановить выполнение программы. Эта команда в действительности и выполняется при выборе позиции **Suspend** после нажатия клавиши **Esc**. Команда **SUSPEND** часто используется как средство отладки. После приостановки можно выполнить требуемые команды с клавиатуры или вызвать их из буфера клавиатуры **HISTORY** и затем продолжить выполнение программы с помощью команды **RESUME**. Созданные во время остановки программы переменные имеют статус **PRIVATE**. После приостановки программы можно не только выполнить команды с клавиатуры но и воспользоваться всеми возможностями УЦ, например для создания файлов БД, ввода и редактирования в

них данных, с использованием сервисных функций УЦ, а также для создания любых компонентов прикладной программы, включая экранные формы ввода и вывода информации, запросы и т.д. Из командного режима можно перейти в УЦ, набрав команду **ASSIST** или нажав функциональную клавишу **F2**, за которой по умолчанию закреплена эта команда. Кроме того, выход на различные экраны УЦ можно осуществить из программы соответствующими командами **CREATE/MODIFY**. Эти команды в основном были описаны в предыдущих разделах за исключением команды

CREATE/MODIFY APPLICATION <**имя программы**>/**?**

которая передает управление на экран генератора прикладных программ. После входа в генератор с целью создания новой программы посредством выбора позиции <**create**> представляется возможность разрабатывать ее самостоятельно, используя текстовый редактор (позиция **dBASE program** в высвечиваемом меню) или с помощью генератора (позиция **Applicatoins Generator**) (более подробно см. разд. 2.6). Как и в других подобных командах, для создания новой или модификации существующей прикладной программы нужно указать ее имя. Можно также выбрать имя существующей программы из списка, высвечиваемого выполнением данной команды с опцией **?**. Причем, если каталог открыт, то высвечиваемый список содержит имена программ из этого каталога, а при закрытом каталоге - имена всех таких программ, хранящихся на диске. Напомним, что прикладные программы, созданные с помощью генератора, имеют расширение **.app** в отличие от обычных программ, которые по умолчанию сохраняются с расширением **.prg**. При сохранении созданной прикладной программы ее имя автоматически заносится в открытый каталог, тогда как компоненты этой программы заносятся только в текущий директорий.

Следует отметить, что при эксплуатации разработанной прикладной программы в ряде случаев нежелательно, чтобы конечный пользователь этой программы имел доступ к средствам разработки прикладных программ. Для этого можно установить команду **SET DESIGN OFF** (по умолчанию **ON**), которая выключает возможность перехода на экраны разработки файлов БД, отчетов, форм, марок, запросов и прикладных программ из командного режима или из УЦ.

В ходе приостановки выполнения программы нельзя модифицировать эту же программу или какую-либо другую открытую программу или процедуру. В этом случае высвечивается сообщение **File already open** (файл уже открыт). Модификацию программы в ходе ее выполнения можно осуществить на экране отладчика, используя его опцию **Edit**. Выбор позиции **Cancel** в меню, высвечиваемом после нажатия клавиши **Esc**, означает инициирование команды **CANCEL**, в результате чего

прекращается выполнение всех программ и процедур, а программные файлы, кроме того, закрываются. Процедурный файл, из которого вызываются в программе процедуры, нужно закрыть командой **CLOSE PROCEDURE**. При попытке возвращения в программу посредством команды **DO <имя программы>** в режиме ввода команд с клавиатуры может не хватить памяти для повторного размещение программного модуля. В этом случае высвечивается меню для выполнения команд **RESUME**, **CANCEL** и **QUIT**. Поэтому лучше возобновлять выполнение программы командой **RESUME**, чем запускать ее заново. Можно также использовать команды **CANCEL** и **QUIT**. Если во время прерывания исполнения программы выполняется обновление файлов БД и затем с клавиатуры вводится команда **ROLLBACK** (восстановить файл в исходное состояние, см. разд. 3.7), то команда **RESUME** возобновляет выполнение программы на строке, следующей после команды **END TRANSACTION** в программе. Команда **RESUME** также возобновляет работу после прерывания командой вызова отладчика **DEBUG** (описание этой команды см. ниже в данном разделе).

Рекомендуется перед командой **RESUME** выполнять команду очистки экрана **CLEAR** (более подробно об этой команде см. разд. 3.4). Это гарантирует, что команды, введенные при остановке программы, не повлияют на последующий вывод информации на экран.

Как уже упоминалось, командой **DO** в процедуру можно передавать из вызывающей программы и обратно параметры, которые в самой процедуре определяются командой

PARAMETERS <список параметров>

Эта команда должна быть первой исполняемой командой в теле процедуры. В качестве параметра в списке может быть любое допустимое выражение, определяющее переменную, в которую передаются данные из соответствующего параметра вызывающей программы. Если эта переменная не объявлена глобальной (**PUBLIC**), то автоматически она имеет статус локальной (**PRIVATE**). После завершения работы процедуры и возвращения в вызывающую программу, локальные переменные процедуры уничтожаются. Типы переменных (выражений) в вызывающей программе и процедуре должны совпадать. Всего можно передать в процедуру до 50 параметров.

Пример:

```
PROCEDURE Ploshad
PARAMETERS Dlina, Shirina, A
A = Dlina * Shirina     && A - площадь
RETURN
DO Ploshad WITH 6,4,A
? A
```

Для передачи параметров, а также и для других целей, например, с целью хранения промежуточных результатов операций, ввода с клавиатуры данных и т.д., в процедурах часто используются переменные и их разновидность - элементы массивов. Как уже упоминалось, переменные могут быть локальными и глобальными. Переменные первого вида существуют только во время работы процедуры или программы, в которых они определены. После завершения работы процедуры (программы) эти переменные уничтожаются. Определение переменных как локальных выполняется командой

PRIVATE <список переменных>
/[ALL [LIKE/EXCEPT <шаблон>]]

Локальные переменные могут иметь одинаковые имена с глобальными переменными, не влияя на последние, поскольку занимают в оперативной памяти разные участки. Между одноименными локальными переменными в разных процедурах также нет взаимодействия. Если в процедуре в качестве локальной объявляется системная переменная, которая по определению всегда глобальная, то создается копия этой переменной со статусом локальной и этой копии присваивается текущее значение системной переменной. Команда **PRIVATE** позволяет объявить со статусом локальной заданный список переменных, а также переменные, имена которых соответствуют заданному шаблону (опция **ALL LIKE**) или за исключением имен, указанных в шаблоне (опция **EXCEPT**). Напомним, что шаблон имени обычно включает общую часть имен и знаки замещения ? и *, первый из которых замещает один алфавитно-цифровой символ, а второй - любое их число. В шаблон можно включать только одни знаки замещения. Это позволяет, например, знаком * задавать все имена, а несколькими знаками ? имена соответствующей длины.

Глобальные переменные можно использовать повсюду в программе и в вызываемых ею процедурах. Для объявления переменных и элементов массива глобальными используется команда

PUBLIC <список переменных>/
[ARRAY <список элементов массива>

Кроме переменных в программах и процедурах можно использовать одно- или двумерные массивы. Каждый элемент массива представляет собой обычную переменную. Представление переменных в виде массива позволяет значительно расширить диапазон применения переменных, а также, в ряде случаев, облегчить реализацию функций, присущих переменным. Перед использованием массивов сначала их надо объявить командой

DECLARE <имя массива 1>
[{<количество строк>,} {<количество колонок>}]
{,<имя массива 2>
[{<количество строк>,} {<количество колонок>}]...}

В синтаксисе этой команды опции указываются фигурными скобками вместо обычно принятых квадратных. При записи команды в программе в квадратные скобки заключаются параметры (размерность) массива - число строк и столбцов. Если указано одно число, то массив одномерный. Формально двумерный массив с одним из параметров, равным единице, фактически является одномерным: **DECLARE [12,1]** или **DECLARE [1,25]**.

Массив, объявленный в программном файле, по умолчанию имеет статус **PRIVATE**, а в командном режиме - **PUBLIC**. Имя массива может быть длиной до 10 символов и подчиняется правилам образования имен переменных и полей файлов БД - должно включать только буквенно-цифровые символы и символ подчеркивания, не содержать пробелов и начинаться с буквы.

Нежелательно использовать имена из одной буквы, так как буквы от **A** до **J** обозначают алиасы рабочих областей. Например, удобно использовать мнемонические имена, описывающие содержимое переменной, или начинать имя с буквы **M**. Все элементы только что объявленного массива содержат логическое значение **.F.**. Тип элемента массива, как и переменной, определяется типом введенных данных.

Не рекомендуется массивам присваивать имена уже имеющихся переменных, так как в этом случае никаких сообщений не выдается и результат обращения к таким массивам непредсказуемый. Обращение к массиву может быть в форме обращения к одному элементу или ко всему массиву в целом. В первом случае использование в командах массива полностью аналогично использованию переменной.

Вторая форма обращения применяется в командах типа **COPY TO ARRAY** или **APPEND FROM ARRAY**. При ссылке на элементы массива, которые не существуют, высвечивается сообщение **Bad array dimension(s)** (ошибка в размерности), а при ссылке на несуществующий массив - **Not an array** (массив не существует).

Примеры:

```
DECLARE Mas [8,4]        && Двумерный массив
DECLARE Mas1 [10]        && Одномерный массив
STORE {15/10/89} TO Dat1 [4,4].]
```

После объявления массива в него можно вводить данные. Ввод данных в переменную или массив (или их изменение) можно осуществить различными способами: в программе, с

клавиатуры или из файла БД. Для ввода данных в переменную или элементы массива в программе используется команда

STORE <выражение> TO <список переменных>
/<список элементов массива>

или альтернативная форма

<переменная>/<элемент массива> = <выражение>

Команда **STORE** позволяет занести одно значение в несколько переменных (элементов массива). Альтернативная форма короче и традиционно используется для числовых переменных, но значение заносится только в одну переменную. В dBASE IV, как и в других языках высокого уровня, значение числовой переменной можно прибавлять (вычитать) само к себе, а значение символьной переменной может быть сцеплено само с собой.

Максимальное число переменных (элементов массива), которые можно использовать в программе, определяется значениями параметров **MVBLKSIZE** и **MVMAXBLKS** файла **Config.db**. Если **MVBLKSIZE = 50**, а **MVMAXBLKS = 10** (задаются по умолчанию), то можно создать до 500 переменных. Имя каждого массива при объявлении занимает один участок памяти, отводимый под переменную. Под элементы массива память распределяется динамически.

Во многих случаях удобно использовать одно имя для переменной и поля файла БД, например при засылке значений полей в соответствующие переменные с целью их промежуточного хранения или обработки.

В командах dBASE IV поле файла БД с именем, совпадающим с именем переменной, имеет приоритет во всех операциях, кроме функции макроподстановки **&**. Как уже указывалось, для установления в этом случае приоритета переменной нужно впереди ее имени поставить символы **M->**, например, **M->Fam**.

Примеры:

```
STORE .F. TO False && Занесение в переменную False значения .F.
STORE 0 TO a,b,c   && Занесение значения 0 в переменные A, B и C
Newdate = {31/10/89}
Mx = 335
```

После завершения работы программы и выхода в dBASE IV все переменные и массивы уничтожаются. Для сохранения их на диске с целью последующего использования применяется команда

SAVE TO <имя файла> [ALL LIKE/EXCEPT <шаблон>]

Эта команда записывает в файл на диске (по умолчанию с расширением **.mem**) все или часть (определяемую заданным

шаблоном), находящихся в текущий момент в оперативной памяти переменных и элементов массивов.

Примеры:

```
SAVE TO B:Mem && Сохраняет в файле Mem все
*  переменные и массивы из программы.
SAVE ALL LIKE d* TO B:Mem && Сохраняет переменные
*  (массивы) с именами, начинающимися с буквы d
SAVE ALL EXCEPT ???X* TO Mem.OLD && Сохраняет переменные (массивы),
*  за исключением, тех, чьи имена содержат в 4-й позиции букву X
```

Для восстановления в оперативной памяти сохраненных ранее переменных и элементов массивов используется команда

RESTORE FROM <имя файла> [ADDITIVE]

По умолчанию предполагается, что файл переменных имеет расширение **.mem**. При этом все текущие значения восстанавливаемых переменных удаляются, если не указана опция **ADDITIVE**. Если эта опция указана, то текущие значения сохраняются и к ним добавляются значения, считанные из файла на диске. Переменные восстанавливаются со статусом **PRIVATE**, если в программе они не были определены как **PUBLIC**. Если же команда **RESTORE** выполняется с клавиатуры, то все переменные восстановливаются со статусом **PUBLIC**.

В dBASE IV можно использовать до 25 000 переменных. Фактически разрешенное число устанавливается как параметр в файле **Config.db** (по умолчанию 500). Реальное число переменных ограничивается объемом свободной оперативной памяти.

Пример:

```
DO Initvar
RETURN
PROCEDURE Initvar
PUBLIC Page, Otvet
Page = 1
Otvet = 'Y'
RETURN
```

Для ускорения работы с внешними процедурами можно использовать команду

SET PROCEDURE TO [<имя файла процедур>]

которая открывает заданный файл процедур. Компилированные в dBASE IV процедурные файлы имеют расширение **.prc**, а некомпилированные - **.prg**. Если не задано расширение процедурного файла, то предполагается, что это файл объектных (компилированных) модулей с расширением **.dbo**. В случае отсутствия и этого файла отыскивается файл с заданным именем среди некомпилированных программ, который перед выполнением компилируется. Команды **SET PROCEDURE TO <Enter>** или

CLOSE PROCEDURE закрывают текущий файл процедур. В каждый текущий момент может быть открыт только один файл процедур.

Пример:

```
<команды>  && Главная процедура
. . .
DO Proc1
<команды>
DO Proc2
<команды>
RETURN
PROCEDURE Proc1
@ 15,0 SAY 'Система готова к работе'
RETURN
PROCEDURE Proc2
@ 10,0 CLEAR
@ 15,10 SAY 'Принтер не готов к работе'
@ 17,10 SAY 'Включите принтер'
RETURN
```

Для выполнения процедуры из файла процедур **Fproc1.prg** нужно выполнить команды

```
SET PROCEDURE TO Fproc1
DO Proc1
CLOSE PROCEDURE
```

Кроме процедур, в программах dBASE IV можно вызывать любые исполнимые двоичные модули. Существуют два способа выполнения внешних модулей. В первом способе нужно предварительно загрузить этот модуль в память командой **LOAD <имя двоичного файла>**. Одновременно можно загрузить до 16 таких модулей (если нет ограничения на объем свободной оперативной памяти) размером до 32 000 байт. По умолчанию для этих модулей предполагается расширение **.bin** (для определения количества и имен загруженных модулей можно использовать команду **LIST/DISPLAY STATUS**). Затем модуль запускается командой

CALL <имя модуля> [WITH <список выражений>]

где имя модуля указывается без расширения.

При вызове в двоичный модуль можно передать параметры, которые определены опцией **WITH**, включающие символьные выражения, поля файла БД, переменные или элементы массива. Список может содержать до семи выражений. Все символьные переменные и выражения должны завершаться нулем (ASCII кодом нуля). При вызове модуля регистр CS указывает на начало модуля в памяти, а регистры DS и BX содержат адрес первого байта переменной, поля, элемента массива или символьного выражения.

Если двоичный модуль формируется из ассемблерной программы, то необходимо соблюдать следующие требования:

первая исполнимая команда этого модуля должна размещаться со смещением нуль;

вызываемый модуль не должен использовать память за пределами размеров самого модуля;

в модуле нельзя изменять размеры переменных, передаваемых в качестве аргументов командой **CALL**;

перед возвращением в dBASE IV модуль должен восстановить регистры CS и SS;

возврат в dBASE IV должен быть выполнен командой ret far.

Вместо команды для запуска загруженного модуля можно использовать функцию

CALL(<выр.C>,<выр.C>/<список имен переменных>)

в которой первое символьное выражение в списке аргументов задает имя модуля (без расширения), а второе символьное выражение (или список переменных) позволяет передать и возвратить результат выполнения модуля как функции. При выполнении такой функции адрес первого аргумента загружается в регистры DS и BX и управление передается в начало загруженного модуля для его выполнения. Переменные или элементы массивов, используемые в качестве передаваемых параметров, должны быть предварительно определены. Можно передавать до семи параметров.

Модули, которые завершаются командой выхода в DOS, должны вызываться командой **RUN/! <имя модуля или команды DOS>**, которая реализует второй способ запуска внешних модулей (а также и выполнение команд DOS). В данном случае загрузку модуля в оперативную память осуществляет DOS.

При выполнении команды **RUN/!** требуется дополнительная емкость оперативной памяти для файла **Command.com** DOS, а также соответствующая емкость памяти для выполнения модуля. Если памяти не хватает, то высвечивается сообщение **Insufficient memory** (недостаточно памяти).

Вызываемые модули и файл **Command.com** должны быть в текущем директории или же их маршрут по директориям следует указать в виде значения параметра **comspec** команды **set** DOS, например, **set comspec = C:\Command.com**. При выполнении команд DOS, особенно в командном режиме, для сокращения записи лучше использовать альтернативную запись команды **RUN** - !. При этом надо иметь в виду, что программные модули некоторых команд DOS, например, **assigne** и **print**, остаются в памяти после их выполнения, поэтому их лучше запускать до загрузки dBASE IV. Перед загрузкой в память внешних модулей целесообразно оценить емкость свободной

оперативной памяти командой **DISPLAY MEMORY**, которая выводит информацию об использовании оперативной памяти (более подробно об этой команде см. разд. 3.2). Если памяти не хватает, то можно удалить из нее ряд неиспользуемых программных модулей, переменных, окон, экранных форм и т.д. Это можно сделать командами

RELEASE <список переменных>
или
RELEASE [ALL [LIKE/EXCEPT <шаблон>]
или
RELEASE [MODULE <список имен модулей>]
/MENUS [<список имен горизонтальных меню>]
/POPUPS [<список имен вертикальных меню>]
/WINDOWS [<список имен окон>]

которые удаляют переменные, загруженные ассемблерные программы, горизонтальные и вертикальные меню, окна и освобождают занимаемую ими память.

Команда **RELEASE ALL** в режиме набора с клавиатуры удаляет все переменные (указанные в шаблоне с опцией **LIKE** или за исключением указаных в шаблоне с опцией **EXCEPT**). Системные переменные не удаляются. При использовании этих команд в программе удаляются только локальные переменные, созданные в текущей программе или в программе нижнего уровня (переменные, созданные программами более высокого уровня, не удаляются).

Команда **RELEASE MODULE** удаляет из памяти модуль, загруженный с помощью команды **LOAD**.

Команда **RELEASE MENUS/POPUPS/WINDOWS** удаляет с экрана и из памяти перечисленные в списке горизонтальные, вертикальные меню и окна. Команды **ON SELECTION, ON PAD** и **ON SELECTION POPUP**, связанные с меню, освобождаются перед удалением меню. Меню можно удалить только после их деактивизации. Если список в команде не указан, то удаляются все меню и окна.

Примеры:

```
RESTORE FROM Mem          && Удаляет все переменные,
RELEASE ALL LIKE m*       && начинающиеся с буквы m
DISPLAY MEMORY

RESTORE FROM               && Удаляет все переменные, за исключением
RELEASE ALL EXCEPT ??X*  && тех, где в третьей позиции буква "X"
DISPLAY MEMORY

RELEASE POPUPS Ed,Ad,Del,Ex && Удаляет вертикальные меню с именами
DISPLAY MEMORY               && Ed,Ad,Del,Ex
```

Для прекращения выполнения любого программного файла используется команда **CANCEL**. Одновременно закрываются все открытые программные файлы и управление возвращается в режим ввода команд с клавиатуры. Процедурные файлы не закрываются. Эта команда часто используется при отладке программ.

Пример:

```
IF Merror   && Если возникла ошибка (значение
   CANCEL   && переменной Merror = .T.), то
ENDIF       && выполнение программы прерывается
```

В программах (и процедурах) кроме процедур можно вызывать функции, созданные пользователем. Функция строится подобно процедуре, но должна начинаться командой **FUNCTION <имя функции>**. Так же, как и процедура, функция завершается командой **RETURN <выражение>**, которая служит для завершения функции и возврата в указанном выражении результата ее работы. Если функция не определяет никакого результата, то возвращается логическое значение **.F.** В функцию, как и в процедуру, можно передавать параметры, которые в функции также должны быть определены командой **PARAMATERS**. В отличие от процедуры при вызове функции передаваемые параметры указываются в скобках после имени функции. Функции, определенные пользователями, употребляются только вместе с командами, например в составе выражений, и вызываются по имени, указанном в команде **FUNCTION**. Имя функции подчиняется требованиям, установленным для имен процедур и файлов (должно включать не более 8 буквенно-цифровых символов и начинаться с буквы), и не должно совпадать с именами команд или системных функций. Перед выполнением функция должна быть откомпилирована и может храниться как отдельный программный файл (открытый или закрытый) с расширением **.dbo** или входить в состав процедурного файла (в этом случае для ее вызова нужно открыть процедурный файл командой **SET PROCEDURE**). Имена функций, используемых в программном файле, содержатся вместе с процедурами в списке, размещаемом в начале объектного модуля этого файла. Поиск функции перед ее выполнением осуществляется аналогично процедурам.

Пример:

```
FUNCTION Stnalog
PARAMETERS Mcena,Mproc
Mstoim = (Mcena*Mproc)+Mcena
RETURN (Mstoim)
Mproc = .10
Mcena = 100
Tstoim=Stnalog(Mcena,Mproc)+Dcost
```

Если переменная Mstoim определена как глобальная, то ее значение можно после вызова функции использовать повсюду в программе в качестве возвращенного функцией значения. В силу специфики выполнения функций, определенных пользователями, в них запрещается использовать ряд команд dBASE IV:

APPEND
APPEND FROM
APPEND FROM ARRAY
APPEND MEMO
ASSIST
BEGIN TRANSACTION/
  END TRANSACTION
BROWSE
CANCEL
CHANGE
CLEAR ALL/FIELDS
CLOSE
CLOSE ALTERNATE
CLOSE FORMAT
CLOSE PROCEDURE
COMPILE
CONVERT
COPY
COPY FILE
COPY INDEXES
COPY MEMO
COPY STRUCTURE
COPY STRUCTURE EXTENDED
COPY TAG
COPY TO ARRAY
CREATE или MODIFY STRUCTURE
CREATE FROM
CREATE VIEW FROM ENVIRONMENT
CREATE/MODIFY APPLICATION
CREATE/MODIFY LABEL
CREATE/MODIFY QUERY/VIEW
CREATE/MODIFY REPORT
CREATE/MODIFY SCREEN
DEBUG
DEFINE BAR
DEFINE BOX
DEFINE PAD
DEFINE POPUP
DEFINE WINDOW
DELETE TAG
DIR
EDIT
ERASE или DELETE FILE
EXPORT
HELP
IMPORT
INDEX
INSERT
JOIN
LABEL FORM
LOAD
LOGOUT
MODIFY COMMAND/FILE
MOVE WINDOW
ON ERROR/ESCAPE/KEY
ON PAD
ON PAGE
ON READERROR
ON SELECTION PAD
ON SELECTION POPUP
PACK
PRINTJOB/ENDPRINTJOB
PROTECT
QUIT
REINDEX
REPORT FORM
RESTORE
RESTORE MACROS
RESTORE WINDOW
RESUME
ROLLBACK
SAVE
SAVE MACROS
SAVE WINDOW
SET
SORT
SUSPEND
TYPE
UPDATE
ZAP

Кроме этих команд, в функциях, определенных пользователями, запрещено использовать функцию макроподстановки **&** и следующие команды **SET**:

CATALOG	FORMAT	SKIP	TRAP
DEBUG	PROCEDURE	SQL	VIEW
DEVICE	RELATION	STEP	WINDOW
FIELDS			

Удобство эксплуатации прикладных программ в настоящее время обычно обеспечивается созданием "дружественного" интерфейса пользователя, важным компонентом которого является создание меню прикладной задачи. Широкое распространение получили меню горизонтального и вертикального типа и их сочетания. С помощью комбинации этих типов можно организовать вложенную древовидную структуру меню, в которой каждая позиция вышестоящего меню служит для раскрытия меню нижнего уровня, а позиция меню самого нижнего уровня выполняет определенные действия.

Чтобы организовать в dBASE IV систему меню, сначала надо определить его отдельные компоненты - меню вертикального и горизонтального типа. Для этого используются специальные команды, начинающиеся словом **DEFINE** - определить. Определение горизонтального меню выполняется командой

DEFINE MENU <имя меню> [MESSAGE <выр.С>]

Эта команда еще не создает горизонтальное меню, а только назначает ему имя и, если требуется, сообщение, которое будет высвечиваться в нижней строке экрана (строке сообщений). Для определения позиций горизонтального меню используется команда

DEFINE PAD <имя позиции>
OF <имя меню> PROMPT <выр.С>
[AT <строка>,<колонка>] [MESSAGE <выр.С>]

которая определяет одну позицию горизонтального меню. Для задания нескольких позиций данная команда повторяется соответствующее число раз. Имя позиции должно быть уникальным и подчиняется обычным правилам написания имен полей и алиасов. В опции **OF** указывается имя ранее определенного меню.

Опция **PROMPT** определяет текст, высвечиваемый с заданной позиции горизонтального меню, определяемой опцией **AT**, в которой указывается номер строки и колонки начала позиции. Данное меню условно называется горизонтальным в отличие от меню вертикального типа, однако команда по его созданию позволяет расположить позиции меню в любом месте экрана

и, таким образом, создать любую форму, в том числе горизонтального и вертикального типа. Если координаты в опции **AT** не задаются, то первая позиция меню помещается в левом верхнем углу экрана, а следующая через пробел от предыдущей позиции на этой же строке.

При размещении горизонтального меню в верхней строке экрана целесообразно установить состояние **SET SCOREBOARD OFF**, чтобы не перекрывать меню информацией о состоянии клавиатуры. Передвижение по позициям меню осуществляется навигационными клавишами.

Опция **MESSAGE** позволяет каждой позиции меню назначить текст, который будет появляться в строке сообщений при выборе данной позиции. Это сообщение перекрывает сообщение, назначенное в команде **DEFINE MENU**.

Вертикальное меню определяется командой

```
DEFINE POPUP <имя меню>
        FROM <строка 1>,<колонка 1>
        [TO <строка 2>,<колонка 2>]
        [PROMPT FIELD <имя поля>]
        /PROMPT FILES [LIKE <шаблон>]
        /PROMPT STRUCTURE]
        [MESSAGE <выр.C>]
```

которая устанавливает его имя (записывается по тем же правилам, что и для горизонтального меню) и координаты окна меню на экране. Опции **FROM** и **TO** определяют координаты левого верхнего и правого нижнего углов окна меню. Если опция **TO** опущена, то высота и ширина окна определяются автоматически в соответствии с количеством позиций и максимальной длиной поля позиции меню. Окно может располагаться в колонках 0-79 и строках 0-23 экрана (если строка состояния не используется). Минимальный размер окна - одна позиция экрана на пересечении колонки и строки. При выводе окна на экран оно закрывает любой текст, включая строку состояния. В каждый момент активным может быть только одно окно, поэтому разрешается определять несколько окон на одном месте экрана. Удаление окна вертикального меню с экрана выполняется с помощью его деактивизирования (более подробно об этом см. ниже в настоящем разделе). Данная команда позволяет определить не только вертикальное меню, но еще и списки (подробно о списках см. в разделе 2.9). Здесь напомним, что существуют списки трех типов: файлов, имен полей и значений полей. Для определения списка файлов используется опция **PROMPT FILES**, которая высвечивает в позициях меню имена файлов из каталога в соответствии с заданным в опции **LIKE** шаблоном. Без этой опции высвечиваются имена всех типов файлов текущего каталога. Опция **PROMPT STRUCTURE** высвечивает в позициях меню все имена полей

активного файла БД или поля, указанные командой **SET FIELDS**, если установлено состояние **SET FIELDS ON**. Опция **PROMPT FIELD** высвечивает в позициях меню содержимое указанного поля (кроме полей **Memo**) для всех записей файла БД. Напомним, что для файлов в других рабочих областях перед именем поля нужно помещать алиас файла БД. Опция **MESSAGE** позволяет высветить текстовое сообщение длиной 79 символов в нижней строке экрана после активизации меню. Это сообщение имеет приоритет перед командой **SET MESSAGE TO**. Назначение текста и других атрибутов одной позиции ранее определенного вертикального меню осуществляется командой

DEFINE BAR <номер строки> OF <имя меню>
PROMPT <выр.C>
[MESSAGE <выр.C>] [SKIP [FOR <условие>]]

Чтобы определить нескольких позиций, нужно эту команду повторить соответствующее число раз. Позиции (строки) вертикального меню нумеруются целыми положительными числами сверху вниз. Позиции меню, которые не помещаются в ранее определенное окно, высвечиваются навигационными клавишами **Вверх** и **Вниз** (вертикальная прокрутка), а записываемый в позиции меню текст, который не помещается в окно, усекается справа. Текст позиции записывается в опции **PROMPT**. В опции **MESSAGE** записывается текстовое сообщение, высвечиваемое в нижней строке экрана после установки курсора на позицию меню. Опция **SKIP** используется для пропуска позиции (высвечивание без выполнения действий), а опция **FOR** задает условие для ее пропуска.

После того как определены позиции меню, обычно требуется назначить им определенные действия, которые будут выполняться при их выборе. Эти действия для позиций горизонтального меню назначаются командой

ON SELECTION PAD <имя позиции горизонтального меню>
OF <имя горизонтального меню> [<команда>]

которая выполняет заданную команду при выборе указанной позиции горизонтального меню. В качестве команды может использоваться любая команда dBASE IV, в том числе команды вызова программ и процедур. Вызванные процедуры могут включать, например, такие функции, как **MENU()** или **PAD()**, для определения имени активного горизонтального меню и его выбранной позиции, чтобы выполнять соответствующие действия. Функция **MENU()** возвращает имя активного меню или пустую строку, если активного меню нет. Эта функция полезна при поиске имени активного меню в случае выхода из него, например, клавишей **F1** для просмотра справочной информации. Функция **PAD()** возвращает имя последней выбранной позиции горизонтального меню.

Напомним, что выбрать позицию меню - это значит установить на эту позицию курсор и нажать клавишу **Enter**. После выполнения команд, вызванных программ и процедур горизонтальное меню опять становится доступным для последующего выбора. Если команда не указывается, то никаких действий при выборе позиции не выполняется. При организации системы меню часто требуется, чтобы при выборе позиции горизонтального меню (которое обычно бывает основным) вызывалось вертикальное меню (прикрепленное к позиции горизонтального меню). Это можно сделать командой

ON PAD <имя позиции горизонтального меню>
OF <имя горизонтального меню>
[ACTIVATE POPUP <имя вертикального меню>]

которая назначает вызов и активизацию вертикального меню при выборе позиции горизонтального меню, одновременно сохраняя на экране горизонтальное меню. Активизация меню означает, что навигационными клавишами можно передвигаться по позициям меню. Прикрепив к позициям горизонтального меню вертикальные меню, получаем систему меню, в которой при движении навигационными клавишами **Вправо** и **Влево** по позициям горизонтального меню вызываются прикрепленные к ним вертикальные меню. Если не указана опция **ACTIVATE POPUP**, то никаких действий при выборе позиций не выполняется. Если позиции горизонтального меню раньше было назначено действие командой **ON SELECTION PAD**, то прикреплять вертикальное меню уже нельзя.

Для назначения действий позициям вертикального меню используется команда

ON SELECTION POPUP <имя позиции вертикального меню>
/ALL [<команда>]

которая выполняет заданную команду или процедуру при выборе позиции вертикального меню. Если команда не указана, то ни одно действие при выборе позиции не выполняется. При использовании опции **ALL** указанная команда выполняется для всех ранее определенных вертикальных меню.

После выбора позиции начинается выполнение назначенных действий, а меню деактивизируется до завершения выполнения этих действий. Обычно выполняемая процедура содержит команду **DO CASE**, позволяющую выполнить операции, соответствующие выбранной позиции вертикального меню. Для идентификации этой позиции могут использоваться функции **POPUP()**, **PROMPT()** и **BAR()**.

Функция **BAR()** возвращает номер последней выбранной позиции вертикального меню или списка. Эта функция возвращает нуль, если в текущий момент нет активного меню или вообще не определено ни одно вертикальное меню или список,

а также, если была нажата клавиша **Esc**, которая деактивизировала меню.

Функция **POPUP()** возвращает имя активного вертикального меню или списка. Если нет активного меню, то возвращается нулевая (нулевой длины) строка.

Функция **PROMPT()** возвращает символьную строку, представляющую собой текст последней выбранной позиции горизонтального, вертикального меню или списка. Если нет активного меню, то возвращается нулевая строка. Для списка файлов возвращается полное имя файла, включающее маршрут по директориям и расширение прописными буквами, для списка полей - имя поля прописными буквами, а для списка значений поля - его содержимое из файла БД.

Обычно для назначения позициям меню каких-либо действий используется выход на процедуру, в которой помещаются команды для выполнения этих действий. В такой процедуре чаще всего применяется команда

```
DO CASE
   CASE <условие>
   <команды>
   [CASE <условие>]
   <команды>
[OTHERWISE]
   <команды>
ENDCASE
```

которая позволяет разветвить процесс выполнения команд в зависимости от выбранной позиции меню. Для этого за ключевым словом **CASE** помещается условие типа **BAR() = I**, где **I** - номер выбранной позиции меню. При удовлетворении этого условия выполняются последующие команды до ключевых слов **CASE**, **ENDCASE** или **OTHERWISE**. Переход на команды после ключевого слова **OTHERWISE** осуществляется в том случае, если условия всех ключевых слов **CASE** были оценены как ложные. После выполнения этих команд другие ключевые слова **CASE** в команде пропускаются и управление и передается на команду, следующую за словом **ENDCASE**, которое завершает данную команду. Естественно, что данная команда имеет более широкое применение и с ключевым словом **CASE** может использоваться любое разрешенное логическое выражение.

Можно формировать вложенные структуры этой команды, т.е. внутри команды после ключевого слова **CASE** помещать другую команду **DO CASE**, а также любые другие команды ветвления и выполнения циклов, например **IF...ENDIF**, **DO WHILE...ENDDO** и др.

Пример:

```
DO CASE
```

```
    CASE Val > 1000
    <команды>
    CASE Val > 100
    <команды>
    CASE Val > 10
    <команды>
    OTHERWISE
    <команды>
ENDCASE
```

Если требуется разветвить процесс только на два направления, можно использовать команду

IF <условие>
  <команды>
[ELSE]
  <команды>
ENDIF

В случае оценки заданного условия как **"Истина"**, т.е. при возвращении логического значения **.T.** выполняются последующие команды. При оценке условия как **"Ложь"**, т.е. в случае возвращения логического значения **.F.** и наличия опции **ELSE**, выполняются последующие команды, а без этой опции управление передается на первую следующую за словом **ENDIF** команду. Данная команда позволяет формировать вложенные структуры любых команд ветвления, в том числе включать и команды **IF...ENDIF**.

При реализации операций, назначаемых позициям меню, нередко требуется организовать циклы повторяющихся действий, например организовать обработку файла БД и др. Это можно сделать с помощью ряда команд и их комбинаций. Одной из таких является команда

DO WHILE <условие>
  <команды>
[EXIT]
  <команды>
[LOOP]
  <команды>
ENDDO

которая обеспечивает повторение последовательности команд, заключенной между словами **DO WHILE** и **ENDDO** до тех пор, пока заданное условие истинно. Если встречается слово **LOOP**, то управление передается в начало цикла команд, а при встрече слова **EXIT** осуществляется выход из команды и управление передается на первую команду после слова **ENDDO**, завершающего данную команду. Обычно выход на слова **LOOP** и **EXIT** реализуется командами ветвления, например командой **IF...ENDIF**.

Действия, аналогичные предыдущей команде, можно выполнить командой

SCAN [<границы>] [FOR <условие>] [WHILE <условие>]
[<команды>]
[LOOP]
[EXIT]
ENDSCAN

Эта команда также предназначена для организации повторяющихся циклов операций с помощью группы команд, расположенных между словами **SCAN** и **ENDSCAN**, но имеет более широкое применение, например при обработке активного файла БД, позволяет осуществлять отбор записей в соответствии с заданными границами и условием в опции **FOR**.

Ключевые слова **LOOP** и **EXIT** в данной команде выполняют те же функции, что и в команде **DO WHILE...ENDDO** - первое передает управление в начало команды на повторение цикла, а второе обеспечивает выход на первую команду после завершающего слова **ENDSCAN**. Повторение циклов осуществляется до тех пор, пока условие, заданное опцией **WHILE**, истинно или указатель записи (при обработке файла БД) достигает конца файла (последней записи, определенной границами). При этом передвижение указателя записей имеет следующие особенности. Указатель устанавливается в начало файла (на первую запись в индексе или в положение, определенное границами или опциями **FOR** и **WHILE**) и передвигается по отобранным записям. После завершения команды указатель остается на последней отобранной записи.

Пример:

```
USE Fio
SCAN FOR .NOT. Otd = '01'
   DO Change WITH Nomer,Fam,Name
ENDSCAN
```

При использовании команды **DO WHILE** предыдущий пример выглядит так:

```
USE Fio
LOCATE FOR .NOT. Otd = '01'
DO WHILE .NOT. EOF()
   DO Change WITH Nomer,Fam,Name
   CONTINUE
ENDDO
```

Предыдущие команды позволяют определить меню и атрибуты их позиций. Для того чтобы активизировать горизонтальное и вертикальное меню (а также списки) используются команды

ACTIVATE MENU <имя меню> [PAD <имя позиции>] и

ACTIVATE POPUP <имя меню>

которые вызывают на экран предварительно определенные меню. Если указано имя позиции горизонтального меню, то курсор устанавливается на эту позицию. В противном случае курсор устанавливается на первую позицию меню. В каждый текущий момент активным является только последнее активизированное меню. Если одно меню активизирует другое, то первое меню приостанавливает свое действие до тех пор, пока второе не будет деактивизировано.

Деактивизация меню выполняется посредством активизации другого меню, нажатия клавиши **Esc** или выполнения команд **DEACTIVATE MENU** и **DEACTIVATE POPUP**. Эти команды деактивизируют активное в текущий момент горизонтальное или вертикальное меню, удаляют его с экрана, оставляя в памяти, и используются только в программах. Если для горизонтального меню установлена команда **ON PAD**, то все связанные с ним вертикальные меню также деактивизируются. Команда деактивизации вертикального меню может использоваться в команде **ON SELECTION** или в процедурах, вызываемых этой командой, что позволяет удалить меню с экрана после того, как осуществлены требуемые действия.

После выполнения команд деактивизации меню управление передается на следующую строку программы.

Для контроля внешнего вида разрабатываемых меню в программах можно использовать команды

SHOW MENU <имя горизонтального меню>
[PAD <имя позиции горизонтального меню>] и
SHOW POPUP <имя вертикального меню>

которые высвечивают меню на экране без активизации.

Команды по созданию меню должны располагаться в следующей последовательности:

1. **DEFINE MENU, DEFINE POPUPS** - определение меню;
2. **ON SELECTION PAD, ON PAD, ON SELECTION POPUP** - назначение команд, выполняемых при выборе позиции;
3. **ACTIVATE MENU, ACTIVATE POPUP** - активизация и высвечивание меню экране.

Пример:

```
DEFINE MENU Test
DEFINE PAD Dir OF Test PROMPT "Директорий" AT 0,0
DEFINE PAD Dec OF Test PROMPT "Удалить меню" AT 0,15
ON SELECTION PAD Dir OF Test DIR
ON SELECTION PAD Dec OF Test DEACTIVATE MENU
ACTIVATE MENU Test
```

В этом примере создается горизонтальное меню с именем Test с позициями "Директорий" и "Удалить меню". Первой из этих позиций назначается команда **DIR**, а второй - команда деактивизации меню.

Пример:

```
Mdel = .F.
DEFINE POPUP View FROM 1,1 TO 6,22
DEFINE BAR 1 OF View PROMPT "Добавить запись"
DEFINE BAR 2 OF View PROMPT "Редактировать запись"
DEFINE BAR 3 OF View PROMPT REPLICATE("-",20);
                     SKIP
DEFINE BAR 4 OF View PROMPT "Удалить запись";
                     SKIP FOR Mdel
ACTIVATE POPUP View
```

Здесь создается вертикальное меню View с позициями "Добавить запись", "Редактировать запись" и "Удалить запись". Последняя позиция пропускается, если переменная Mdel имеет логическое значение **.T.**

Перед последней позицией рисуется горизонтальная линия, которая является незадействованной (пропускаемой) позицией.

Пример:

```
ON SELECTION POPUP Edit DO Edit
PROCEDURE Edit
DO CASE
   CASE BAR() = 1
   APPEND BLANK
   EDIT NEXT 1
   CASE BAR() = 2
   EDIT
   CASE BAR() = 3
   DELETE
ENDCASE
RETURN
```

В данном примере демонстрируется разветвление выполнения команд в зависимости от выбранной позиции меню, номер которой возвращается функцией **BAR()**.

Ниже приводятся примеры создания горизонтального меню с прикрепленными к каждой его позиции вертикальными меню.

Пример:

```
DEFINE MENU Main
DEFINE PAD View OF Main PROMPT "Добавить" AT 0,0
DEFINE PAD Go   OF Main PROMPT "Перейти " AT 0,11
DEFINE PAD Pri  OF Main PROMPT "Печать  " AT 0,19
DEFINE PAD Exit OF Main PROMPT "Выход   " AT 0,27
```

Определяется горизонтальное меню и четыре его позиции с именами View, Go, Pri и Exit.

Пример:

```
DEFINE POPUP Viewpop FROM 1,1 TO 6,18
DEFINE POPUP Gopop   FROM 1,20 TO 8,37
DEFINE POPUP Pripop FROM 1,40 TO 4,60
DEFINE POPUP Exitpop FROM 1,60 TO 2,78
```

Определяются четыре вертикальных меню с именами View, Go, Pri и Exit.

```
ON PAD View OF Main ACTIVATE POPUP Viewpop
ON PAD Go   OF Main ACTIVATE POPUP Gopop
ON PAD Pri  OF Main ACTIVATE POPUP Pripop
ON PAD Exit OF Main ACTIVATE POPUP Exitpop
```

Затем позициям горизонтального меню назначается активизация вертикальных меню.

```
DEFINE POPUP Viewpop FROM 1,1 TO 6,22
DEFINE BAR 1 OF Viewpop PROMPT "Пустую запись"
DEFINE BAR 2 OF Viewpop PROMPT "В конец БД"
DEFINE BAR 3 OF Viewpop PROMPT "Из файла БД"
```

После этого определяются позиции вертикального меню Viewpop. Аналогично определяются другие вертикальные меню.

```
ACTIVATE MENU Main
```

Наконец, активизируется вертикальное меню, к позициям которого прикреплены вертикальные меню.

Ниже приводится пример создания вертикального меню с двумя позициями. Затем это меню активизируется и после выбора первой позиции осуществляется выход в DOS, а вторая позиция деактивизирут меню и передает управление в dBASE IV (в УЦ или в командный режим).

Пример:

```
DEFINE POPUP Exit FROM 3,30
DEFINE BAR 1 OF Exit PROMPT "Выход в DOS"
DEFINE BAR 2 OF Exit PROMPT "Выход в  dBASE IV"
ACTIVATE POPUP Exit
ON SELECTION POPUP Exit DO Ex
PROCEDURE Ex
DO CASE
   CASE BAR() = 1
   QUIT
   CASE BAR() = 2
   DEACTIVATE POPUP
ENDCASE
```

Для создания удобства конечным пользователям, которые будут эксплуатировать разрабатываемую прикладную программу, обычно на экран выводится справочная информация о работе

программы, действиях пользователя и т.д. Примером такой информации служат сообщения, которые высвечиваются в строке сообщений при выборе позиций меню. В ряде случаев возникает необходимость вывода на экран больших объемов справочной информации. Это можно сделать различными способами, в частности командой

TEXT
<текст>
ENDTEXT

которая выводит на экран (или на принтер, если установлено состояние **SET PRINTER ON**) текст точно так, как он записан между словами **TEXT** и **ENDTEXT**. Символ **&** в тексте интерпретируется как обычный символ, а не как функция макроподстановки.

В гл. 2 было описано распространенное средство обеспечения удобства работы пользователя - создание и использование макроопределений, представляющих собой имитацию последовательности нажатий клавиш на клавиатуре. В программе можно предусмотреть выход в УЦ для создания макроопределения. После создания макроопределения для его повторного выполнения обычно макроопределение записывают в библиотеку на диске. Если предполагается использовать созданное макроопределение при эксплуатации разрабатываемой прикладной программы, то в ряде случаев полезно записать его в собственный библиотечный файл макроопределений. Такую операцию выполняет команда **SAVE MACROS TO <имя файла макроопределений>**, которая записывает созданное макроопределение в библиотечный файл на диске с расширением **.key**. Можно задать другое расширение, указывая его затем при загрузке файла библиотеки в оперативную память перед выполнением макроопределения, осуществляемом командой **RESTORE MACROS**. Напомним, что в качестве имен макроопределений могут использоваться комбинации клавиш от **Alt-F1** до **Alt-F9** и **Alt-F10** с последующей буквой от **A** до **Z**. Таким образом, всего можно задать 35 уникальных имен макроопределений. В каждый библиотечный файл можно записать от 1 до 35 макроопределений.

Выполнение загруженного в память макроопределения осуществляется командой **PLAY MACRO <имя макроопределения>**. Имя макроопределения подчиняется правилам образования имен полей файла БД и переменных (может состоять из десяти буквенно-цифровых символов, начинаться с буквы и не должно содержать пробела). Запуск команды выполнения макроопределения в программе отличается от запуска в командном режиме. В последнем случае команда запускается на выполнение сразу после нажатия клавиши **Enter**, а в первом только после инициирования другой команды. Поэтому несколь-

ко последовательно записанных в программе команд **PLAY MACRO** выполняются в обратной последовательности: первой выполняется последняя записанная команда, второй - предпоследняя и т.д.

Командный язык dBASE IV включает достаточно средств для обеспечения качественной отладки создаваемых прикладных программ. Одним из таких средств является отладчик программ, вызываемый командой

DEBUG <имя файла>/<имя процедуры>
[WITH <список параметров>]

Эта команда, как и команда **DO**, выполняет программный или процедурный модуль, но в среде отладчика, имеющего свой экран. Экран отладчика содержит четыре окна: для запуска программы (процедуры) с любой точки и просмотра команд при их выполнении (окно отладки), редактирования программы (окно редактирования), установки точек прерывания выполнения программы (окно прерываний) и вывода результатов раскрытия выражений при выполнении команд (окно результатов). В окне отладки, находящемся в нижней части экрана, высвечивается текущая рабочая область, имя файла БД, имя программы и процедуры, номер записи файла БД, номер строки программы (процедуры), главный (управляющий) индекс и приглашение **ACTION** для ввода команды в ходе процесса отладки. Переход к этому приглашению выполняется клавишами **Esc** или **Ctrl-End** из любого места экрана отладчика. Команды задаются условными буквами, описание которых в любой момент можно высветить в правой верхней части экрана при нажатии функциональной клавиши **F1** вызова справочной информации. Ниже приводится перечень условных букв и выполняемых ими действий:

E - вызывает окно редактирования программы с текстовым редактором dBASE IV. После редактирования нужно сохранить программный файл, чтобы избежать потери сделанных изменений. Следует отметить, что после возвращения в отладчик продолжается выполнение старого программного файла. Для запуска отредактированного программного файла нужно снова инициировать отладчик, при этом корректно завершив предыдущий сеанс работы с отладчиком, т.е. буквой **Q** в приглашении **ACTION**;

B - активизирует окно прерываний, находящееся с правой стороны в середине экрана. В это окно можно ввести с клавиатуры несколько точек прерывания выполнения программы в виде условий, которые проверяются после выполнения каждой командной строки. Если хотя бы одно из этих условий выполняется (возвращается логическое значение **.T.**), то программа приостанавливается и высвечивается экран отладчика;

D - активизирует окно результатов, находящееся в левой стороне средней части экрана. В каждую строку этого окна можно ввести с клавиатуры любое выражение, после чего справа высвечивается его результат, например значение переменной, поля и т.д.;

L - определяет, какую следующую строку программы нужно выполнить;

N - выполняет следующую за текущей команду в программе и передает управление на приглашение **ACTION**. Если в текущей программе (процедуре) командой **DO** вызывается другая процедура, то она выполняется вне среды отладчика, хотя точки прерывания остаются в силе. Если перед буквой **N** указать число, то перед выходом на приглашение **ACTION** будет выполнено заданное число команд;

P - высвечивает трассировку выполняемой программы, включая имя текущей программы, процедуры и номер выполняемой строки;

Q - осуществляет выход из отладчика и прекращение выполнения программы;

R - запускает программу на выполнение до точки прерывания или появления ошибки;

S - выполняет следующую команду, после чего управление передается на приглашение **ACTION**. Если перед буквой **S** указать число, то перед выходом на приглашение **ACTION** будет выполнено заданное число команд. В отличие от буквы **N** здесь процедуры, вызываемые из текущей программы, выполняются в среде отладчика;

X - осуществляет выход из программы и передает управление в командный режим, в котором можно выполнить любые команды. Возврат в отладчик осуществляется командой **RESUME**. Из режима ввода команд с клавиатуры можно также войти в УЦ клавишей **F2** или командой **ASSIST** и после выполнения требуемых операций вернуться обратно с помощью позиции **Exit to dot prompt** меню **Exit** или клавишей **Esc**.

Нажатие клавиши **Enter** в приглашении **ACTION** вызывает выполнение операций **1S** или **1N** в зависимости от того, какая из команд **S** или **N** последний раз была выполнена. По умолчанию выполняется команда **1S**. Нажатие клавиши **F1** в любом месте экрана отладчика высвечивает, а при повторном нажатии удаляет панель со справочной информацией о выполняемых приглашением **ACTION** командах. Нажатие клавиши **F9** удаляет окна отладчика с экрана и высвечивает информацию, которая была скрыта экраном отладчика.

Опция **WITH** позволяет передать в отлаживаемую программу параметры, как и в команде **DO**. Список параметров может содержать любые разрешенные в dBASE IV выражения. Однако отлаживаемая программа в среде отладчика может принять не

более восьми констант, определяющих имена файлов. Число передаваемых переменных не ограничено.

Помимо отладчика для определения имени программы или процедуры, в которой произошла ошибка, можно использовать функцию **PROGRAMM()**. Эта функция возвращает имя программы или процедуры (без расширения), а не файла процедур. В командном режиме эта функция действует только во время приостановки выполнения программы. Например, в программу можно включить такую команду локализации ошибки:

ON ERROR ? "Ошибка в " + PROGRAM() + ;

"Строка " + STR(LTRIM(LINENO()))

В число средств отладки программ также входит команда **SET TRAP ON** (**OFF** - по умолчанию), которая устанавливает состояние, автоматически вызывающее отладчик при обнаружении в программе ошибки. Программа приостанавливается на строке, вызвавшей ошибочную ситуацию. Кроме того, в этом состоянии в любой момент можно вызвать отладчик клавишей **Esc**. Следует отметить исключительную полезность команды **SET TRAP** при отладке сложных программ. Опыт отладки программ на языке dBASE IV показывает, что при возникновении ошибочной ситуации нередко портятся открытые файлы и возникают другие непредсказуемые эффекты. Команда **SET TRAP** вылавливает ошибку и, по существу, локализует ее, не давая последствиям ошибки распространится на данные и программу. Из опыта отладки программ можно также порекомендовать не оставлять без надобности открытыми файлы и немедленно закрывать их после окончания работы с ними. Если одновременно установлено состояние **ON ERROR**, то оно имеет приоритет перед **ON TRAP** и при обнаружении ошибочной ситуации выполняется обработка этой ситуации командой **ON ERROR**. В том случае, когда не установлено ни состояние **ON ERROR** и ни **SET TRAP ON**, ошибочная ситуация вызывает приостановку программы и высвечивается такое же меню, как и при нажатии клавиши **Esc**, предлагающее прекратить выполнение программы (**C**), игнорировать ошибку (**I**), приостановить программу с выходом в командный режим (**S**). Напомним, что возврат из этого режима на продолжение программы осуществляется командой **RESUME**.

Весьма полезной при отладке программ является команда **SET ECHO ON** (**OFF** - по умолчанию), выводящая на экран (или принтер) выполняемые команды (в программах или в командном режиме). При работе в среде отладчика данная команда позволяет наблюдать выполнение всех отладочных операций, включая выполнение команд программы. Однако высвечивание этих операций в ходе отладки на экране может помешать выводу другой информации, например меню,

экранных форм, данных из файлов БД и т.д. В этом случае можно направить информацию об отладочных операциях на принтер, если установить состояния **SET DEBUG ON** (**OFF** задается по умолчанию) и **SET PRINTER ON**. Команда **SET ECHO ON** позволяет высветить только выполняемые команды, но не их результаты. Для высвечивания результатов служит команда **SET TALK ON** (устанавливается по умолчанию), которая позволяет выводить номера записей файлов БД, значения переменных, результаты выполнения команд **APPEND FROM**, **COPY**, **PACK**, **STORE** и **SUM**. Если установлено состояние **SET TALK OFF**, то не выводятся ни результаты операций, ни сообщения об ошибках. Если требуется результаты направить на принтер, то как и для информации об отладочных операциях надо установить команды **SET DEBUG ON** и **SET PRINTER ON**.

В отладчике предусмотрены различные команды для организации процесса отладки, в том числе для обеспечения различных режимов выполнения программы - по одной строке, несколько командных строк, до точки прерывания и т.д. Режим выполнения программы по одной строке помимо отладчика можно также установить командой **SET STEP ON** (**OFF** задается по умолчанию). При этом высвечивается результат выполнения командной строки. Перед переходом на следующую строку высвечивается предложение

Press SPACE to Step, S to Suspend, or Esc to Cancel...

В качестве ответа можно нажать клавишу Пробел для следующего шага, букву **S** для приостановки программы и перехода в командный режим или клавишу **Esc** для прекращения выполнения программы.

К средствам отладки можно также отнести команду **SET HELP ON** (устанавливается по умолчанию), которая высвечивает сообщение об ошибке в команде, выполняемой в командном режиме, и предлагает на выбор прекратить ее выполнение (**CANCEL**), отредактировать команду (**EDIT**) или получить по ней справочную информацию. Эту информацию о любой команде или функции можно также высветить командой **HELP [<ключевое слово>]**, в которой в качестве ключевого слова задается имя команды, функции или раздела справочной информации. Команда без ключевого слова высвечивает содержание справочной информации, что эквивалентно нажатию клавиши **F1**.

Тип монитора обычно устанавливается как параметр в файле **Config.db**. Однако для конкретной конфигурации можно задать тип монитора и число строк экрана командой

SET DISPLAY TO MONO/COLOR/EGA25/EGA43/MONO43

В командном языке dBASE IV имеется несколько команд и функций, играющих вспомогательную роль. В их число входит

команда **SET HOURS TO [12/24]**, которая устанавливает время, высвечиваемое на экране ПК в 12- или 24-часовом формате. По умолчанию устанавливается 12-часовой формат. Опытным пользователям полезны функции **OS()** и **VERSION()**, возвращающие имя операционной системы и номер версии dBASE IV, для учета специфики установленной операционной среды в разрабатываемых прикладных программах.

3.7. ВЫЧИСЛЕНИЯ И ОБРАБОТКА СТРОК

В предыдущих разделах описаны основные команды и функции по созданию файлов БД и манипуляции данными. С их помощью можно организовать любые вычисления.

Однако в языке dBASE IV имеется ряд команд и функций, специально предназначенных для выполнения стандартных вычислительных операций над данными. Например, для вычисления среднего арифметического значения числовых полей файла БД можно использовать команду

AVERAGE [<список выр.N>] [<диапазон>]
[FOR<условие>] [WHILE<условие>]
[TO<список переменных>/
TO ARRAY<имя массива>]

Как это видно из синтаксиса команды, можно вычислять среднее арифметическое не только отдельных полей, но и любых разрешенных выражений с числовыми данными, в которые могут входить различные поля, переменные, функции, связанные операторами (см. определение выражения в разд. 3.1). Напомним, что опции **FOR** и **WHILE** определяют условия отбора записей из заданного диапазона.

Результаты усреднения записываются в указанные в списке переменные или элементы одномерного массива в порядке, определяемом списком выражений. Если не указан список выражений, определяющих усредняемые данные, то по умолчанию усредняются все числовые поля.

Если установлено состояние **SET TALK ON**, то результаты выводятся на экран. Такой вывод можно использовать, если не требуется заносить результаты в переменные или массив. Если переменных или элементов массива больше, чем выражений в списке, то оставшиеся после записи переменные или элементы массива сохраняют свои значения. В противном случае оставшиеся не записанными результаты усреднения игнорируются.

Пример:

```
USE Sotr
* Вычисляется средняя зарплата сотрудников 1 отдела
AVERAGE Zarpl FOR Otd="01" TO Sredzar
```

Для выполнения стандартных статистических расчетов можно использовать команду

CALCULATE [диапазон] <список опций>
[FOR <условие>] [WHILE<условие>]
[TO <список переменных> /
TO ARRAY <имя массива>]

В состоянии **SET TALK ON** результаты выводятся на экран. Если при этом также установлено состояние **SET HEADING ON**, то выводятся имена полей и функций. Опция **TO** позволяет записать результаты вычисления функций в переменные или массив.

Статистическая операция задается в списке опций в виде функций с аргументами, которые определяют поле файла БД:

AVG(<выражение>) - вычисляется среднее арифметическое поля данных;

CNT() - подсчитывается количество записей в файле БД, удовлетворяющих условию, заданному опцией **FOR**, т.е. когда условие оценивается как **.T.**;

MAX(<выражение>) - определяется наибольшее по величине значение числовых полей, полей типа даты и символьных полей;

MIN(<выражение>) - определяется наименьшее значение аналогично предыдущей функции;

NPV(<процент>,<поступления>,<начальное значение>) - вычисляется чистое текущее значение накопленной суммы денежных средств, где

<процент> - процент, представленный десятичной дробью;

<поступления> - серия денежных поступлений или платежей (числа со знаками + или -), записанных в поле данных, которое может быть представлено выражением;

<начальное значение> - числовое выражение, представляющее начальное значение вложенных денежных средств, которое записывается со знаком "минус", так как является платежем;

STD(<выражение>) - вычисляется среднеквадратическое отклонение значений поля данных;

SUM(<выражение>) - вычисляется сумма значений поля данных;

VAR(<выражение>) - вычисляется дисперсия значений поля данных.

Пример:

```
USE Finans
CALCULATE FOR Otd='01' SUM(Pay), CNT() TO Totpay, Kol
? "Количество сотрудников="+LTRIM(STR(Kol,3))+;
  "Выплаченная сумма="+LTRIM(STR(Totpay,8,2))
```

В число статистических функций предыдущей команды входит одна, связанная с финансовыми расчетами. Кроме того,

ряд стандартных финансовых расчетов можно выполнить с помощью специальных функций. Так, функция

FV(<депозит>,<процент>,<число периодов>)

позволяет вычислить общую сумму равных регулярных депозитов в течение заданного числа периодов времени при фиксированном проценте, где **<депозит>** - числовое выражение, значение которого может быть положительным или отрицательным числом; **<процент>** - процентная ставка каждого периода, выражаемая положительной десятичной дробью; **<число периодов>** - числовое выражение, представляющее количество платежных периодов.

Для вычисления постоянных регулярных платежей, требуемых для погашения кредита с постоянным процентом в течение заданного числа платежных периодов, используется функция

PAYMENT(<кредит>,<процент>,<число периодов>)

где **<кредит>** - числовое выражение, представляющее сумму кредита (может быть положительным или отрицательным числом); **<процент>** - процентная ставка каждого периода, выражаемая положительной десятичной дробью; **<число периодов>** - числовое выражение, представляющее количество платежных периодов.

Для вычисления суммы вложенных денежных средств используется функция

PV(<платежи>,<процент>,<число периодов>)

где **<платежи>** - числовое выражение, представляющее постоянные регулярные платежи, которые могут быть положительным или отрицательным числом; **<процент>** - процентная ставка каждого периода, выражаемая положительной десятичной дробью; **<число периодов>** - числовое выражение, представляющее количество платежных периодов.

Кроме команды **CALCULATE** количество записей файла БД в указанных границах, удовлетворяющих заданным условиям, можно подсчитать и записать в переменную командой

COUNT [TO <переменная> [<границы>]
[FOR <условие>]
[WHILE <условие>]

Пример:

```
USE Pokupki
COUNT FOR LIKE('03/??/89',DTOC(Date_pokup)) TO Mart_zak
? "Количество заказов в марте=";
  +LTRIM(STR(Mart_zak,3,0))
```

Суммирование данных можно выполнить командой

SUM [<границы>] [<список выражений>]

**[TO <список переменных>
/TO ARRAY <имя массива>]
[WHILE <условие>] [FOR <условие>]**

Размещение результатов и другие варианты использования данной команды аналогичны командам **AVERAGE** и **CALCULATE**. Результатом суммирования является число типа **F**.

Пример:

```
USE Finans
DECLARE Msum [2]
SUM Oklad, Doplata TO ARRAY Msum
? "Сумма основного оклада="+STR(Msum [1],3,0)
? "Сумма доплат="+STR(Msum [2],3,0)
```

Одной из типовых вычислительных операций является подсчет подытогов, выполняемый с помощью команды

**TOTAL ON <ключевое поле> TO <имя файла>
[<границы>] [FIELDS <список полей>]
[FOR <условие>] [WHILE <условие>]**

которая вычисляет подытоги числовых полей записей (всех полей или указанных в списке полей) с одинаковыми значениями заданного ключевого поля активного файла БД и записывает результаты в новый файл. По умолчанию расширение нового файла имеет вид **.dbf**. Активный файл БД должен быть проиндексирован или отсортирован по ключевому полю. Новый файл БД имеет ту же структуру, что и суммируемый файл (исключая поля **Memo**). Числовые поля нового файла содержат подытоги, а другие - значение соответствующего поля первой записи из группы с одинаковым значением ключевого поля. Если размер поля в новом файле слишком мал для занесения результата суммирования, то в качестве значения поля помещаются знаки *, обозначающие переполнение.

Кроме описанных команд и функций в языке dBASE IV имеется достаточно большой набор математических функций, функций обработки символьных данных и данных типа даты, а также ряд вспомогательных функций, позволяющих выполнять широкий спектр действий по обработке данных.

Рассмотрим сначала математические функции. К их числу относится вычисление квадратного корня **SQRT(<выр.N>)** и абсолютного значения **ABS(<выр.N>)** числового выражения, экспоненты **EXP(<выр.N>)**, десятичного **LOG10(<выр.N>)** и натурального **LOG(<выр.N>)** логарифмов. Функция **PI()** возвращает число "пи". В группу тригонометрических функций входит синус **SIN(<выр.N>)**, косинус **COS(<выр.N>)** и тангенс **TAN(<выр.N>)** угла, заданного в радианах. Результат этих тригонометрических функций возвращается в виде числа типа **F**.

Обратные тригонометрические функции представлены функциями арксинус **ASIN(<выр.N>)**, арккосинус **ACOS(<выр.N>)**, арктангенс **ATAN(<выр.N>)** и функцией **ATN2(<выр.N1>,<выр.N2>)**, которая вычисляет функцию арктангенс для заданного значения синуса **(выр.N1)** и косинуса **(выр.N2)**. Все обратные тригонометрические функции возвращают результат в радианах. Для перевода градусов в радианы используется функция **DTOR(<выр.N>)**, а обратно (из радиан в градусы) - **RTOD(<выр.N>)**.

Функция **CEILING(<выр.N>)** вычисляет наименьшее целое, которое больше или равно заданному числовому выражению, а **FLOOR(<выр.N>)** возвращает наибольшее целое число, которое меньше или равно заданному числовому выражению. Для округления чисел можно использовать функцию **ROUND(<выр.N1>,<выр.N2>)**, в которой первое выражение задает округляемое число, а второе - количество оставляемых дробных десятичных позиций. При задании в последнем выражении отрицательного числа округляется соответствующее число позиций целой части числа.

Функция **INT(<выр.N>)** преобразует числовое выражение в целое число посредством отбрасывания дробной части. Для возвращения большего или меньшего из двух чисел, дат или символьных выражений применяются соответственно функции

MAX(<выр.N1>/<выр.D1>,<выр.N2>/<выр.D2>)

и

MIN(<выр.N1>/<выр.D1>,<выр.N2>/<выр.D2>)

Остаток от деления двух числовых выражений возвращается функцией **MOD(<выр.N1>,<выр.N2>)**. Предусмотрена также функция **RAND([<выр.N>])** для вычисления случайного числа в диапазоне от **0** до **0.999999**, в которой числовое выражение определяет базу для вычисления следующего случайного числа. Если задано отрицательное число, то в качестве базы выбирается системное время. В качестве базы по умолчанию принято число **100001**. Знак числа можно определить функцией **SIGN(<выр.N>)**, которая возвращает **1**, если числовое выражение положительное, **-1** - если отрицательное и **0** - для нуля.

Для обработки символьных данных в языке dBASE IV предусмотрена довольна большая группа функций. Нередко при обработке строк требуется найти начальную позицию слова или его части в тексте. Для этого используется функция **AT(<выр.C>,<выр.C>/<имя Memo поля>)**, которая возвращает число, представляющее собой начальную позицию первой символьной строки во второй символьной строке или в поле **Memo**. Функция **LEFT(<выр.C>/<имя Memo поля>,<выр.N>)** позволяет извлечь заданное число символов из символьного выражения или поля **Memo**, начиная с первого символа слева,

а функция **RIGHT(<выр.C>/<переменная>,<выр.N>)** - из символьного выражения или переменной, начиная с последнего символа справа. Для ввода в строку заданной подстроки и удаления части исходной строки можно использовать функцию

STUFF(<выр.C1>,<выр.N1>,<выр.N2>,<выр.C2>)

в которой первое символьное выражение задает исходную символьную строку или переменную, а первое числовое выражение определяет номер позиции этой строки, с которой вводится подстрока, задаваемая вторым символьным выражением. Если эта позиция равна нулю, то ввод подстроки начинается перед первым символом. Если номер позиции выходит за пределы строки, то ввод начинается после последнего символа. Второе числовое выражение задает число удаляемых символов из исходной строки вправо, начиная с позиции ввода подстроки. Если это число равно нулю, то из исходной строки не удаляется ни одного символа.

Выше были описаны функции извлечения подстроки из символьной строки справа или слева. Похожими возможностями обладает функция

**SUBSTR(<выр.C>/<имя memo поля>,
<начальная позиция>[,<число символов>])**

позволяющая извлечь заданное число символов из символьного выражения или поля **Memo**. Если число символов не задано или же оно больше длины строки, то извлекается подстрока с начальной позиции и до конца строки. Часто при обработке строки требуется определить ее длину. Для этого можно использовать функцию **LEN(<выр.C>/<имя memo поля>)**, которая возвращает длину указанного выражения или содержимого поля **Memo**. Весьма полезными являются функции **LTRIM(<выр.C>)** и **RTRIM(<выр.C>)** или **TRIM(<выр.C>)**, первая из которых удаляет ведущие (левые), а вторая и третья - конечные (правые) пробелы. При поиске данных нередко нужно определять, с какой буквы - строчной или прописной - начинается строка. Для этого применяются функции **ISLOWER(<выр.C>)** и **ISUPPER(<выр.C>)**, возвращающие логическое значение **.T.**, если указанное символьное выражение начинается соответственно со строчной или прописной буквы.

Преобразование строчных букв в прописные и наоборот можно выполнить функциями **UPPER(<выр.C>)** и **LOWER(<выр.C>)**. Для создания строки с повторяющимися подстроками используется функция **REPLICATE(<выр.C>, <выр.N>)**, в которой подстрока указывается символьным, а число повторений - числовым выражением. Строку, состоящую из заданного числа пробелов, можно получить функцией **SPACE(<выр.N>)**. Если требуется сцепить символьные и

числовые данные, то последние нужно преобразовать к символьному типу. Это можно сделать функцией

STR(<выр.N> [,<длина>] [,<число десятичных позиций>])

которая преобразует число в символьную строку. Здесь длина определяет полное число символов в создаваемой функцией строке, включая десятичную точку, знак "минус" и дробные десятичные позиции, число которых определяется отдельной опцией.

Если длина и число дробных десятичных позиций не заданы, то по умолчанию предполагается 10 символов и число округляется до целого. Если длина строки меньше, чем число цифр в целой части числового выражения, то выводится знак *. Для обратного преобразования (из символьного типа в числовой) используется функция **VAL(<выр.C>)**. Если в начале символьного выражения стоит не цифра или не пробел, то возвращается 0. Символы цифр переводятся в числовое значение до тех пор, пока не встретится нецифровой символ. К числу функций обработки строк можно также отнести функции **ASC(<выр.C>)** и **CHR(<выр.N>)**. Функция **ASC(<выр.C>)** возвращает десятичный код ASCII первого символа заданного выражения, а функция **CHR(<выр.N>)** преобразует десятичный код ASCII, заданный числовым выражением, в символ и позволяет вывести на экран или принтер любой символ из расширенного набора IBM. Числовое выражение может включать числа от 1 до 255. Следует отметить, что не все принтеры поддерживают набор символов IBM. Поэтому перед выводом на принтер следует убедиться по его руководству, что предполагаемые для печати символы им поддерживаются. Код ASCII 7, заданный в качестве аргумента функции, инициирует звуковой сигнал. При включении функции с аргументом нуль **CHR(0)** в команду, она должна помещаться слева от оператора отношения, поскольку при выполнении оператора отношения сначала анализируется правая часть и при обнаружении нулевой строки операция прекращается и возвращается логическое значение **.T.**

Примеры:

```
. ? CHR(65)
A

. CHR(7)+"Будьте внимательнее!"
ppp = "Иванов"

? ppp = CHR(0)
.T.

? CHR(0)  = ppp
.F.
```

Ряд числовых функций возвращает результаты в форме с плавающей запятой (тип **F**). Преобразование этой формы в двоично-десятичную форму числа (тип **N**) выполняется функцией **FIXED(<выр.N>)**. Нередко используется и обратное преобразование (из типа **N** в тип **F**) функцией **FLOAT(<выр.N>)**, например для внутреннего представления результатов вычисления, когда получаются очень большие или очень малые числа.

Довольно большая группа функций позволяет обрабатывать данные типа даты. В предыдущих разделах уже упоминалась функция **CTOD(<выр.C>)** для преобразования даты, записанной в символьном виде, во внутреннее представление данных типа даты. Обратное преобразование выполняется функцией **DTOC(<выр.D>)**, а функция **DTOS(<выр.D>)** преобразует выражение типа даты в символьную строку формата CCYYMMDD, где CC - век; YY - год; MM - месяц; DD - день. Такое преобразование выполняется независимо от установки команд **SET CENTURY** и **SET DATE**. Первая команда, например, позволяет вводить дату в символьную строку, последняя - составлять сложное индексное выражение с датой, охватывающей несколько веков.

При выполнении операций с датами нередко требуется получить число, обозначающее порядковый номер дня недели и месяца. Это можно сделать с помощью следующих функций: **DOW(<выр.D>)**, возвращающей номер дня недели (воскресенье = 1), **DAY(<выр.D>)**, возвращающей день месяца из выражения типа даты, **MONTH(<выр.D)**, возвращающей номер месяца в дате. Для возвращения названия дня недели и месяца используются функции **CDOW(<выр.D>)** и **CMONTH(<выр.D>)**. Полное значение года (с цифрами века) возвращается функцией **YEAR(<выр.D>)**. Как уже отмечалось, по умолчанию принят американский формат даты: месяц, число, год. Для преобразования даты в формат вида: число, месяц, год - применяется функция **DMY(<выр.D>)**, а обратное преобразование в американский формат выполняет функция **MDY(<выр.D>)**. Функция **TIME()** возвращает системное время в виде символьной строки в формате ЧЧ:ММ:СС, где ЧЧ - часы; ММ - минуты; СС - секунды.

3.8. РАБОТА В СРЕДЕ ЛВС

Пакет dBASE IV может быть установлен для работы в однопользовательском и многопользовательских режимах (в среде ЛВС). Все программы и файлы, созданные в однопользовательском режиме, можно впоследствии использовать и в многопользовательском режиме, предварительно установив его.

Многопользовательский режим позволяет совместно и зачастую практически одновременно использовать (разделять) программы и файлы БД без потери их целостности, обеспечивая при этом защиту информации.

Однако внешне видимое как одновременное разделение программ и данных на самом деле реализуется последовательно аппаратными и программными средствами ЛВС.

Кроме этих средств в организации процесса разделения файлов обычно участвуют и средства используемых пакетов СУБД. В частности, пакет dBASE IV обеспечивает четыре уровня защиты от одновременного доступа к файлам в среде ЛВС: захват файлов, захват записей, установку файла в исключительное владение одним пользователем и процедуру обновления (транзакции) файла БД.

Целостность файлов БД в среде ЛВС поддерживается специальными командами и функциями. Для ряда команд позволяющих, например, выполнять с клавиатуры ввод и обновление данных, захват файлов и записей осуществляется автоматически, а для других команд режим захвата можно установить специальной командой. При попытке открыть файл, захваченный или установленный для исключительной обработки каким-либо пользователем, в состоянии автоматического повторения попыток захвата появляется сообщение об ошибке: **File is in use by <имя>. Retrying lock, press Esc to cancel.** (Файл захвачен пользователем <имя>. Попытки захвата повторяются, для отмены нажмите клавишу **Esc**.)

При отсутствии состояния автоматического захвата высвечивается сообщение: **File is in use another. Retrying!** (Файл захвачен другим пользователем. Попытайтесь еще !) В этом случае имеется возможность повторить попытку, выбрав позицию **RETRY** меню на экране. Если и после этого файл не открывается, то снова высвечивается то же сообщение. Выбор позиции **ABORT** прекращает попытки открыть файл.

Если пользователь входит в запись разделенного файла БД с целью обновления, то сначала осуществляется попытка захватить эту запись и другие связанные с ней записи. Если эти записи уже захвачены другим пользователем, то в состоянии автоматического повторения захвата появляется сообщение об ошибке: **Record is in use by <имя>. Retrying lock, press Esc to cancel.** (Запись захвачена пользователем **<имя>**. Попытки захвата повторяются, для отмены нажмите клавишу **Esc**) Если захват оказывается успешным, система определяет, были ли выполнены изменения другим пользователем в каком-либо поле после того, как осуществлен доступ к записи. Если изменения были, то они высвечиваются на экране.

При отсутствии состояния автоматического повторения захвата в случае захвата записи другим пользователем появляется другое сообщение: **Record is in use another. Retrying!**

(Запись захвачена другим пользователем. Попытайтесь еще !) В этом случае, как и при захвате файла, имеется возможность повторить попытку, выбрав позицию **RETRY**. Если и после этого захват записи не выполняется, то снова высвечивается то же сообщение. Выбор позиции **ABORT** прекращает запрос на обновление записи.

Для установки режима автоматического захвата записей и файлов БД при выполнении любых команд используется команда **SET LOCK ON** (задается по умолчанию). Захваченный пользователем файл или запись другой пользователь может только читать. После завершения выполнения команды захват файла или записи также автоматически освобождается.

При установке состояния **SET LOCK OFF** автоматический захват отменяется не для всех команд, а только для команд, выполняющих действия в режиме чтения. Для команд, которые имеют как режим чтения, так и записи данных автоматический захват отменяется только для режима чтения. Ниже в табл. 3.9 приводится список команд, которые автоматически захватывают файлы.

Таблица 3.9. **Перечень команд, автоматически захватывающих файлы**

Команда захвата	Выполняемое действие	Уровень	Отмена командой SET LOCK OFF
@GET/READ	Редактирование	Запись	Нет
APPEND FROM	Обновление	Файл	Нет
APPEND [BLANK]	Обновление	Запись	Нет
AVERAGE	Только чтение	Файл	Да
BROWSE	Редактирование	Запись	Нет
CALCULATE	Только чтение	Файл	Да
CHANGE/EDIT	Редактирование	Запись	Нет
COPY TAG/INDEX	Чтение/Запись	Файл	Да (чтение),Нет (запись)
COPY[STRUCTURE]	Чтение/Запись	Файл	Да (чтение),Нет (запись)
COUNT	Только чтение	Файл	Да
DELETE/RECALL	Обновление	Запись	Нет
DELETE/RECALL	Обновление	Файл	Нет
EDIT	Обновление	Запись	Нет
INDEX	Чтение/Запись	Файл	Да (чтение),Нет (запись)
JOIN	Чтение/Запись	Файл	Да (чтение),Нет (запись)
LABEL	Только чтение	Файл	Да
REPLACE	Обновление	Запись	Нет
REPLACE[диапазон]	Обновление	Файл	Нет
REPORT	Только чтение	Файл	Да
SET CATALOG ON	Каталог	Файл	Нет
SORT	Чтение/Запись	Файл	Да (чтение),Нет (запись)
SUM	Только чтение	Файл	Да
TOTAL	Чтение/Запись	Файл	Да (чтение),Нет (запись)
UPDATE	Обновление	Файл	Нет

Команды **BROWSE**, **EDIT** и **CHANGE** обеспечивают автоматический захват только одной, обновляемой в текущий момент записи, тогда как остальные записи доступны для обновления другим пользователям. Захват осуществляется собственно в момент выполнения изменений. После завершения изменений и перехода к следующей записи или файлу захват записи или файла автоматически освобождается и запись или файл могут обновляться другим пользователем. Захваченный файл всегда может просматриваться другим пользователем. Таким образом, в этих командах реализован весьма гибкий метод защиты от одновременного обновления, что позволяет другим пользователям в ЛВС просматривать и редактировать не захваченные записи.

Захват записи в командах осуществляется автоматически при редактировании каждого поля в момент, когда нажимается любая, кроме навигационных, клавиша. Для явного указания захвата записи можно использовать клавиши **Ctrl-O**, а для освобождения от захвата - повторно клавиши **Ctrl-O** или **Ctrl-R**. При передвижении курсора на другую запись захват записи, на которой был курсор, автоматически освобождается за исключением команды **REPLACE**. При выполнении этой команды в случае, когда в качестве диапазона обновляемых записей установлена не одна запись, захват не освобождается до тех пор, пока не будет выполнена другая команда. Если следующей опять выполняется команда **REPLACE**, то запись остается захваченной. Это обеспечивает обновление всех данных одной записи до освобождения ее захвата. В дополнение к автоматическому захвату можно захватывать файл БД в текущей или указанной алиасом рабочей области с помощью функции **FLOCK([алиас])**. В этом случае, как и при автоматическом захвате, другой пользователь может только читать, а не обновлять файл. Обычно такой захват применяется перед выполнением операций с файлом БД как с целым. После успешного захвата эта функция возвращает логическое значение **.T.**, а при попытке захвата уже захваченого файла возвращается логическое значение **.F.** Если захваченный файл связан с другими файлами, то все связанные файлы также захватываются. При освобождении от захвата одного из связанных файлов автоматически освобождаются и другие.

В том случае, когда пользователю нет необходимости захватывать весь файл, можно использовать функцию **RLOCK([<список выр.С>,<алиас>]/[<алиас>])** для захвата одной или нескольких записей. Вместо слова **RLOCK** можно использовать альтернативный глагол **LOCK**. В списке символьных выражений задаются номера захватываемых записей (например, 1,2,5,7,9). Если список не указан, то захватывается текущая запись в файле БД. Задание алиаса позволяет захватывать записи файла БД из другой (не активной в текущий

момент) рабочей области. Напомним, что алиасом может быть номер рабочей области, имя файла БД или косвенная ссылка на файл БД (имя переменной, которая содержит имя файла БД). Если файл БД, в котором захватываются записи, связан с другими (командой **SET RELATION**), то захватываются и все связанные записи, но в количестве не более 50. Для проверки захвата используется эта же функция без аргументов, которая в случае успеха захвата всех записей возвращает логическое значение **.T.** Если хотя бы одна запись из списка не может быть захвачена, то не происходит захвата и других записей и функция **RLOCK()** возвращает логическое значение **.F.** Другие пользователи могут только читать захваченные функцией **RLOCK()** записи. Освобождение захваченных функциями **FLOCK()** и **RLOCK()** записей осуществляется только командой **UNLOCK**, после закрытия файла БД или выхода из dBASE IV.

Для того чтобы обнаружить захват записи другим пользователем, нужно, чтобы при захвате была записана соответствющая информация. Естественно эту информацию помещать в эту же запись. Но для хранения такой информации нужно выделить специальное поле. В dBASE IV добавление этого поля в структуру файла БД выполняется командой **CONVERT [TO <выр.N>]**, которая вводит поле с именем **_dbaselock** в структуру БД и создает новый файл БД с тем же именем и расширением. При этом старый файл остается и ему присваивается расширение **.cvt**. Поле **_dbaselock** содержит информацию, необходимую для обнаружения захвата полей в многопользовательском режиме. Длина этого поля определяется заданным выражением и может быть от 8 до 24 байт (по умолчанию - 16). Это поле резервируется для хранения следующих значений:

счетчика обновлений (**Count**), представляющего двухбайтовое шестнадцатеричное число, используемое функцией **CHANGE()** для обнаружения факта обновления записи;

времени захвата (**Time**), представляющее трехбайтовое шестнадцатеричное число;

даты захвата (**Date**), представляющей трехбайтовое шестнадцатеричное число;

имя пользователя (компьютера) (**Name**), осуществившего захват (длиной до 16 символов).

Счетчик, время и дата размещаются в первых восьми позициях поля **_dbaselock**. Остальную часть поля в зависимости от установленной длины занимает имя пользователя (компьютера), осуществившего захват. Если поле занимает 8 байт, то имя пользователя не вводится. При каждом обновлении записи изменяется значение счетчика поля **_dbaselock**. Функция **CHANGE()** считывает это значение из поля, записанного на диске, и сравнивает со значением счетчика, записанным в системной переменной с этим же именем при считывании записи в память.

Различия этих значений означает, что запись обновлялась другим пользователем, и функция **CHANGE()** возвращает логическое значение **.T.** При перемещении указателя на другую запись возвращается логическое значение **.F.** Команда **GOTO RECNO()** читает заново поле **_dbaselock** текущей записи и последующее выполнение функции **CHANGE()** уже возвращает логическое значение **.F.** до тех пор, пока другой пользователь не обновил запись. Следует отметить, что использование функции **CHANGE()** не всегда правильно фиксирует изменения в записи, хотя счетчик обновлений правильно отражает обновление записи в ходе транзакции. В случае прекращения транзакции и восстановления исходного файла командой **ROLLBACK** этот факт не будет отражен в поле **_dbaselock** и соответственно функция **CHANGE()** будет неправильно отражать ситуацию. Информацию из поля **_dbaselock** можно извлечь функцией **LKSYS(n)**, которая при **n = 0** возвращает время, при **n = 1** дату захвата, а при **n = 2** имя пользователя (компьютера), захватившего запись или файл.

Как уже отмечалось, записи, захваченные функцией **RLOCK()**, нужно освобождать командой **UNLOCK [ALL/IN <алиас>]**, если только вообще файл БД не закрывается или не выполняется выход из dBASE IV. Эта команда освобождает от захвата любым пользователем запись или файл в текущей рабочей области или заданной алиасом, что позволяет другим пользователям модифицировать данные. Если последним захватом был захват записи, то освобождаются все записи. Если же последний захват был захватом файла, то этот файл и освобождается.

Опция **ALL** позволяет освободить все сделанные захваты записей и файлов во всех рабочих областях, а опция **IN** освобождает записи и файлы от захвата в указанной рабочей области.

Пример:

```
USE Fio
IF RLOCK()=.T.
   REPLACE Fam WITH "Петров"
   REPLACE Name WITH "Петр"
   UNLOCK
ENDIF
```

Известно, что факт захвата файла или записей другими пользователями можно установить функциями **FLOCK()** и **RLOCK()**. С целью обнаружения освобождения захвата надо периодически повторять попытки захвата. Для автоматизации этого процесса можно использовать команду **SET REPROCESS TO <выр.N>**, которая устанавливает число (от -1 до 32000, по умолчанию 0), определяющее количество попыток захвата записи или файла в ЛВС перед выводом сообщения об ошибке.

Неограниченное число попыток устанавливается числом -1. Если установлено значение 0 и не установлено состояние **ON ERROR**, то при попытке захвата записи или файла другим пользователем выводится сообщение: **Please Wait, another user has locked this record or file** (Пожалуйста, подождите, другой пользователь захватил запись или файл) и процесс захвата повторяется неограниченно. Для выхода из этого процесса можно использовать клавишу **Esc**. При установке числа больше 0 приведенное выше сообщение не высвечивается.

Команды **BROWSE** и **EDIT**, как уже отмечалось, позволяют одновременно обновлять один файл БД несколькими пользователями. В этом случае важно отслеживать процесс обновления и знать, какие именно записи уже были обновлены. Это можно сделать с помощью команды **SET REFRESH TO <выр.N>**, которая устанавливает интервал времени между проверками файла с целью определения выполненных изменений в файле в среде ЛВС командами **BROWSE** и **EDIT**. При этом файл БД должен быть предварительно реорганизован командой **CONVERT** с целью введения поля **_dbaselock**. Численное выражение задает интервал времени в секундах в диапазоне от 1 до 3600 секунд (по умолчанию 0). По истечении заданного интервала на экран выводятся все обновленные данные. Если интервал равен 0, то проверка файлов не выполняется.

Выше была рассмотрена защита целостности данных в среде ЛВС на уровне захвата файлов и записей, которая дает возможность одному пользователю обновлять, а другим только просматривать разделенные файлы.

Следующий уровень защиты - это установка файла в исключительное распоряжение одним пользователем, так что другие пользователи не имеют никакого доступа к этому файлу, т.е. даже не могут его читать. Такая установка выполняется командой **SET EXCLUSIVE ON** (**OFF** по умолчанию). Команды **CREATE** и **SAVE** автоматически устанавливают состояние **SET EXCLUSIVE ON**.

Кроме описанных в системе обеспечения целостности данных существует специальная команда для организации массового обновления данных (транзакции) как в однопользовательском режиме, так и в среде ЛВС:

BEGIN TRANSACTION [<маршрут по директориям>]
<команды транзакции>
END TRANSACTION

Эта двухфазная команда позволяет начать транзакцию, записать сделанные в файле БД изменения и установить опцию для восстановления исходных значений (**ROLLBACK**). Вторая фаза команды завершает процесс обновлений файла БД и отменяет опцию **ROLLBACK**.

Процесс обновления записей состоит из совокупности элементарных действий, включающих:

добавление новых записей;

изменение данных в записи;

пометку записи для ее последующего удаления.

Первая фаза команды **BEGIN TRANSACTION** начинает записывать эти элементарные действия в создаваемый файл транзакции с именем **Translog.log** в однопользовательской среде и с именем **<имя ПК>.log** в среде ЛВС, где **<имя ПК>** - имя компьютера, с которого стартует команда в ЛВС. В оглавлении файла БД помещается метка, указывающая на то, что файл находится в состоянии изменения. Если требуется отменить изменения и восстановить файл в исходное состояние, то для этого используется созданный файл транзакции. До выполнения команды **END TRASACTION** файл транзакции находится в активном состоянии и позволяет удалить обновления и возвратить файл в исходное состояние с помощью команды **ROLLBACK**. Убедившись, что изменения выполнены правильно, можно перейти к выполнению фазы **END TRANSACTION**. Эта команда деактивизирует транзакцию, удаляет файл транзакции и уничтожает метку состояния обновления файла. После этого для восстановления файла в исходное состояние можно использовать только страховые копии исходного файла. Перед началом новой серии обновлений необходимо завершить предыдущую серию. Следует отметить, что из команд транзакции нельзя создавать вложенные структуры и обе фазы команды должны находиться в одной процедуре.

Команды, которые используются для обновления файлов или изменения параметров среды, также записываются в файл транзакции. В составе этих команд могут быть не только команды выполнения элементарных действия по обновлению, но также команды создания новых файлов БД и каталогов, индексных файлов и т.д. При этом могут применяться любые команды, которые не перезаписывают существующие файлы или не закрывают открытые файлы. Это ограничение требуется по той причине, что в противном случае нельзя будет выполнить команду **ROLLBACK**.

Элементарные действия по обновлению обычно осуществляются следующими командами: **APPEND**, **BROWSE**, **CHANGE**, **DELETE**, **EDIT**, **RECALL**, **REPLACE**, **UPDATE**.

Новые файлы могут создаваться соответствующими командами, при условии, если они не закрывают открытых файлов:

COPY [STRUCTURE] TO <имя файла> [EXTENDED]
CREATE <имя файла> [FROM <имя файла>]
EXPORT TO <имя файла>
IMPORT FROM <имя файла>
INDEX...TO <имя файла>

JOIN...TO <имя файла>
SET CATALOG TO <имя файла>
SORT...TO <имя файла>
TOTAL...TO <имя файла>

Некоторые из этих команд разрешены при создании новых файлов и не разрешены, если они в ходе транзакции закрывают открытые файлы: **CREATE [FROM], IMPORT, INDEX ON, SET CATALOG TO, SET INDEX TO, USE**.

В ходе транзакции не разрешены следующие команды: **CLEAR ALL, CLOSE ALL/DATABASE/INDEX, DELETE FILE, ERASE, INSERT, MODIFY STRUCTURE, PACK, RENAME, ZAP**.

При попытке выполнения этих команд в ходе обновления дается сообщение об ошибке. Вообще для полной гарантии восстановления исходного файла, перед проведением транзакции рекомендуется делать страховую копию.

Пример:

```
PROCEDURE Nakladn
SET REPROCESS TO 15
ON ERROR DO Err_proc
BEGIN TRANSACTION
   USE Nacladn ORDER Schet
   SCAN FOR .NOT. Есть
        .  .  .
        .  .  .
        .  .  .
   ENDSCAN
END TRANSACTION
IF .NOT. ROLLBACK()
   @ 20,1 SAY "Восстановление файла с помощью ;
               команды ROLLBACK не удалось "
   @ 21,1 SAY "Необходимо восстановить файл с ;
               помощью страховой копии"
ENDIF
ON ERROR
RETURN
PROCEDURE Err_proc
otvet = " "
DO CASE
   CASE ERROR() + 108 && Файл захвачен другим пользователем
   @ 20,10 SAY " Файл захвачен другим пользователем"
   @ 21,10 SAY " Попытаться еще раз ? (Да/Нет)" ;
           GET otvet
   READ
   @ 20,10 CLEAR TO 21,79
   IF UPPER (otvet) = "ДА"
      RETRY
   RETURN
   ELSE
```

```
        IF COMPLETED()
           * ---- Ошибок в ходе обновления не было
           @ 20,10 SAY "Файл, захваченный другим пользователем, ;
                        не входит в состав транзакции"
           @ 21,10 SAY "Возврат в главное меню"
           RETURN TO MASTER
        ENDIF
        @ 20,10 SAY "Файл восстанавливается в ;
                     исходное состояние. Ждите ..."
        ROLLBACK
     ENDIF
     RETURN TO MASTER
     CASE ERROR() = 109
        .  .  .  && Аналогичная процедура
  ENDCASE
  RETURN
```

Для проверки успешности завершения транзакции используется функция **COMPLETED()**, которая возвращает логическое значение **.T.**, если транзакция файлов завершена успешно (была выполнена фаза **END TRANSACTION**), и значение **.F.**, если выполнена только первая фаза **BEGIN TRANSACTION**.

В случае прекращения транзакции для восстановления исходных файлов применяется команда **ROLLBACK [<имя файла БД>]**. Кроме файлов БД эта команда восстанавливает индексные файлы в состояние до их обновления, а также закрывает и удаляет все индексные файлы и файлы БД, которые создаются в ходе транзакции. Эта команда помещается между командами начала **BEGIN TRANSACTION** и завершения транзакции **END TRANSACTION**. Восстанавливаемые файлы должны быть открыты, а процесс транзакции должен быть завершен. Если не указывается имя файла БД, то восстанавливаются все обновленные файлы БД, а при указании имени - только этот файл БД.

Файл транзакции включает содержимое каждой записи до и после ее обновления. Команда **ROLLBACK** сравнивает файл транзакции с содержимым активного файла БД, восстанавливает все выполненные изменения и приводит файл БД в исходное состояние. Неуспешное выполнение команды **ROLLBACK** может быть по двум причинам:

нет соответствия между содержимым записей до и после обновления;

файл транзакции не считывается.

После успешного выполнения команды **ROLLBACK** файл транзакции восстанавливается в исходное состояние и остается открытым для обеспечения последующих транзакций.

Пример:

```
ON ERROR ROLLBACK
BEGIN TRANSACTION
```

```
    REPLACE ALL Oklad WITH Oklad*1.1
  END TRANSACTION
```

Проверку успешности выполнения команды **ROLLBACK** можно осуществить с помощью функции **ROLLBACK()**, которая возвращает логическое значение **.T.**, если последняя команда **ROLLBACK** выполнена успешно. В противном случае возвращается **.F.**, до тех пор, пока или команда **ROLLBACK** не будет выполнена успешно или не будет осуществлен выход из dBASE IV.

Как отмечалось выше, команда **ROLLBACK** может не завершиться успешно или может возникнуть необходимость прекратить транзакцию, не прибегая к этой команде. В этом случае в оглавлении файла остается метка, указывающая на то, что файл все еще находится в состоянии обновления и, следовательно, недоступен для обработки. Для удаления этой метки используется команда **RESET [IN <алиас>]**, после выполнения которой файл БД становится доступным для последующего обновления. Эту команду можно использовать только в командном режиме для выхода из необычных ситуаций. Опция **IN** позволяет удалять метку состояния обновления файла БД, находящегося в рабочей области, заданной алиасом.

Для проверки того, находится ли файл БД в состоянии обновления, используется функция **ISMARKED([<алиас>])**, которая возвращает логическое значение **.T.**, если в оглавлении файла БД есть метка о состоянии обновления, и **.F.**, если нет.

К числу средств обеспечения целостности данных можно также отнести команду создания и поддержки защиты информации **PROTECT**. Эта команда, управляемая с помощью меню, используется администратором банка данных для обеспечения защиты информации. Команда может применяться в однопользовательском и многопользовательском режимах и позволяет осуществить защиту информации на трех уровнях:

защиту входа в систему;

защиту файлов и полей данных;

шифрование данных.

Более подробно о системе защиты информации в dBASE IV см. разд. 2.9. Здесь лишь упомянем, что важным элементом системы защиты информации является установка уровней доступа пользователей, которая выполняется администратором системы. Уровни доступа определяют привилегии по использованию данных в файлах БД. Численные значения уровней доступа пользователей применяются в системе защиты информации для определения возможности доступа к тем или иным данным в режиме, управляемом меню, и в программах обработки данных. Численное значение уровня доступа последнего вошедшего в систему пользователя можно определить с помощью функции

ACCESS(). Хотя эти значения хранятся в зашифрованном файле и поэтому недоступны другим пользователям, тем не менее их можно достаточно легко извлечь из некомпилированных программ с расширением **.prg**, в которых используется функция **ACCESS()**. Поэтому при работе в среде ЛВС рекомендуется использовать только компилированные модули, а исходные с расширением **.prg** удалять. Кроме того, целесообразно хранить файл данных о защите системы **Dbsystem.db** в том же директории, что и системные файлы dBASE IV, во избежание его утраты. Если этого файла нет, то экран входа в систему не высвечивается. Для пользователя, вошедшего в систему минуя, экран входа, функция **ACCESS()** возвращает нуль, что не дает ему доступа к зашифрованным файлам. Поэтому в начале программы обработки файлов БД в среде ЛВС рекомендуется проверить уровень доступа каждого вошедшего в систему пользователя и если он равен нулю, то рекомендовать войти повторно или обратиться к администратору системы.

Система защиты информации предусматривает автоматическое шифрование файлов БД при их создании, если установлена команда **SET ENCRYPTION ON**, которая обеспечивает шифрование всех вновь создаваемых файлов БД (следует отметить, что при этом сохраняются и незашифрованные копии, которые надо удалить. Эта процедура описана в разд. 2.9). Однако в ряде случаев, при установленной системе защиты требуется создавать незашифрованные файлы БД или же расшифровывать уже созданные, например, с целью экспорта этих файлов. Установка состояния **SET ENCRYPTION OFF** (задается по умолчанию) позволяет создавать новые незашифрованные файлы при выполнении операций над ранее зашифрованными файлами, например, такими командами, как **COPY**, **JOIN**, **TOTAL** и др.

При работе в среде ЛВС обязательно подключается система защиты информации, хотя могут быть задействованы не все ее уровни. Однако уровень защиты входа в систему действует всегда. Чтобы войти в систему, пользователь набирает с клавиатуры свое системное имя, имя группы, в которой он будет работать, и пароль, которые были предварительно назначены администратором системы. После окончания работы в системе пользователь должен выполнить команду **LOGOUT**, которая фиксирует выход пользователя из системы и высвечивает экран - приглашение для нового входа в систему. Одновременно эта команда закрывает все открытые пользователем файлы БД и связанные с ними файлы, а также программные файлы.

Для идентификации последнего пользователя, вошедшего в систему, используется функция **USER()**, которая возвращает его имя. Полный состав пользователей ЛВС можно установить командой **DISPLAY USERS**, которая выводит имена всех пользователей (компьютеров), работающих в текущий момент в ЛВС,

считывая их из файла **Login.db**, размещенного в текущем директории. Следует отметить, что имена компьютеров должны быть уникальными, так как если в ЛВС работают пользователи с одинаковым именем,-данная команда выводит это имя один раз. Функция **NETWORK()** позволяет определить, работает ли система dBASE IV в среде ЛВС, и возвращает **.T.**, если да, и **.F.**, если нет. Эта функция может использоваться в программах для ветвления процесса выполнения операций в зависимости от того, установлен ли пакет dBASE IV в среде ЛВС или нет.

Глава 4
ЯЗЫК SQL

4.1. ОБЩИЕ СВЕДЕНИЯ О ЯЗЫКЕ SQL

Язык SQL был впервые разработан фирмой IBM и затем одобрен многими компаниями в качестве стандарта языка управления реляционными БД для больших и мини-ЭВМ. В то же время стандартом языка микро-ЭВМ фактически стал язык dBASE. Использование языка SQL в СУБД dBASE IV позволяет связать на логическом уровне программные среды больших и мини-ЭВМ с микро-ЭВМ.

Этот язык ориентирован на операции с данными, представленными в виде логически взаимосвязанной совокупности таблиц, в отличие от языка dBASE, который ориентирован на операции с данными, представленными в виде записей.

Использование концепции операций, ориентированных на табличное представление данных, позволило создать компактный язык SQL с небольшим набором команд. Это дало возможность достаточно легко определять, выводить и обновлять информацию в БД, упростив программирование сложных запросов. Таблица в языке SQL представляет собой совокупность строк и столбцов. В языке dBASE строкам таблицы соответствуют записи, а колонкам - поля файла БД.

Кроме обычных таблиц, язык SQL позволяет создавать особый тип таблиц - выборку, представляющую собой подмножество строк и колонок из одной или нескольких таблиц. Часто выборку называют виртуальной таблицей, поскольку она фактически не содержит данных, а только позволяет их выводить. Данные в выборке отражают их реальное изменение в соответствующих таблицах и, наоборот, изменение данных в обновляемых выборках приводит к изменению этих данных в соответствующих таблицах. Таким образом, языковые средства

SQL в отличие от языка dBASE IV обеспечивают автоматическое обновление связанных файлов БД.

Особенность команд языка SQL состоит в том, что они ориентированы в большей степени на конечный результат обработки данных, чем на процедуру этой обработки. Система определяет оптимальный путь для вывода данных. Такой подход принципиально отличается от других языков управления данными, требующих знание структуры БД и последовательности операций, ведущих к выводу данных.

Полный набор языка SQL включает менее 30 команд, обеспечивающих все операции, необходимые для определения таблиц, выборок, создания запросов на вывод, обновление, удаление и ввод информации. Команды языка SQL легко использовать, так как в них осуществляется ссылка на одни и те же элементы - строки и колонки таблицы или выборки.

Для создания прикладных программ в операционных средах больших и мини-ЭВМ язык SQL обычно используется вместе с языками программирования высокого уровня типа FORTRAN, COBOL, C. В программах dBASE IV также можно комбинировать язык SQL и dBASE IV.

Команды языка dBASE IV позволяют создавать экранные формы ввода и вывода информации, разрабатывать и выводить отчеты, создавать меню и другие элементы прикладных программ, которые нельзя создать средствами языка SQL.

Языковые средства dBASE IV также предназначены для общего управления выполнения прикладной программой. Язык SQL ориентирован в основном на поиск и отбор данных, а также поддержание актуальности БД.

Как и в языке dBASE IV, команды SQL могут выполняться в режиме набора с клавиатуры (командном режиме) и в программах. В командном режиме команды языка SQL, так же как и команды языка dBASE IV, выполняются по одной и результаты выводятся сразу после их выполнения. Выполнение команд dBASE и SQL осуществляется раздельно. Для этого предусмотрены два режима: на языке dBASE (по умолчанию) и на языке SQL. При работе на языке SQL допускается выполнение тех команд и функций dBASE, которые не дублируют соответствующие команды языка SQL или не имеют одинакового наименования.

В прикладных программах команды языка SQL оформляются в виде модулей с расширением **.prs**, а команды языка dBASE в виде модулей с расширением **.prg**. Переключение с одного языка на другой в ходе выполнения прикладной программы осуществляется автоматически в зависимости от исполняемого модуля.

В командном режиме язык SQL устанавливается командой dBASE **SET SQL ON**, а возвращение в dBASE осуществляется командой **SET SQL OFF**. Можно назначить режим SQL по

умолчанию в файле **Config.db**, введя **SQL = ON** или **COMMAND = SET SQL ON**.

Эффективное выполнение команд языка SQL реализуется благодаря хранению специальной ссылочной информации на каждую таблицу и выборку. Эта информация содержится в файлах, называемых каталогами таблиц, которые формируются при создании БД командой SQL **CREATE DATABASE**.

На языке SQL, как и на dBASE IV, обеспечивается однопользовательский и многопользовательский (сетевой) режимы работы. В ходе выполнения ввода, обновления или удаления данных автоматически осуществляется захват данных в таблицах SQL. Если операция не может быть выполнена ввиду захвата данных другим пользователем, целесообразно в программу включать команды повторного захвата. Кроме того, работая на языке SQL в среде ЛВС, можно на языке dBASE выполнять специальные команды по защите файлов БД, например, команды транзакции **BEGIN TRANSACTION** и **END TRANSACTION**. Если операция не завершается успешно, то, как и на языке dBASE IV, можно восстановить БД в исходное состояние с помощью команды **ROLLBACK**.

Команды языка SQL в командном режиме вводятся, как и команды dBASE, в нижней строке экрана после приглашения - слова .SQL - или в окне, которое вызывается нажатием клавиш **Ctrl-Home**. В нижней строке экрана размещается до 254 символов, а в окне при записи в несколько строк до 1024 символов. Команду SQL нужно завершить точкой с запятой и нажать для выполнения клавишу **Enter** при вводе в нижней строке экрана или клавиши **Ctrl-End** при вводе в окне.

Напомним, что точка с запятой используется также в программах на языке dBASE IV для обозначения продолжения команды на следующей строке.

При наборе в окне после нажатия клавиш **Ctrl-End** команда высвечивается в нижней части экрана после слова .SQL и затем выполняется.

Каждая команда языка SQL (называемая также предложением) начинается глаголом, определяющим имя базовой операции. В состав многих команд входят ключевые слова и предложения (команды), уточняющие выполнение базовой операции. Кроме того, в команде SQL нужно определить данные, которые будут обрабатываться (множество строк в одной или нескольких таблицах), и операцию, которую требуется над ними выполнить.

В командном режиме можно получить справочную информацию по использованию команд языка SQL и допустимых команд dBASE IV, набрав команду **HELP**. Высвечивается меню - содержание справочной информации, в котором предлагается выбор тем: **SQL terms** (термины SQL), **Using SQL** (использование SQL), **SQL commands** (команды SQL), **dBASE**

commands in SQL (команды dBASE в SQL), **How to use help** (как пользоваться справочником), **Exiting SQL** (выход из SQL). После выбора темы высвечивается информация, которая может располагаться на нескольких экранах.

Движение вперед по экранам выполняется клавишей **F4**, а назад - клавишей **F3**. Внизу экрана имеется горизонтальное меню с позициями: **CONTENTS** (содержание), **RELATED TOPICS** (связанные темы), **BACKUP** (возвращение) и **PRINT** (вывод на печать).

Можно получить справочную информацию по конкретной команде, если после команды **HELP** набрать глагол этой команды или набрать только глагол команды и нажать клавишу **F1**.

Если выполняемая команда SQL содержит синтаксические ошибки, то высвечивается окно с сообщением об ошибке и горизонтальное меню с позициями: **Cancel** (прекратить выполнение), **Edit** (редактировать команду) и **Help** (справочная информация).

Язык SQL оперирует с понятием БД, которая содержит всю информацию, необходимую для обработки данных в прикладной программе. В состав БД SQL входят следующие компоненты:

таблицы - основные структуры данных в БД;

выборки - тип виртуальной таблицы, которая обеспечивает ввод и вывод определенных строк и колонок из одной или более таблиц;

синонимы - альтернативные имена таблиц и выборок;

индексы - файлы, присоединяемые к таблицам для обеспечения более быстрого поиска данных и поддержки целостности БД;

каталоги - множество таблиц в каждой БД, которые описывают БД и ее содержимое.

Перед созданием или использованием таблиц SQL необходимо создать или активизировать существующую БД. Каждая БД хранится в определенном директории вместе с таблицами и связанными файлами, включая каталоги таблиц. В каталогах таблиц содержится информация о базовых таблицах, выборках, индексах и синонимах, а также данные по защите информации и по операциям, выполненным над таблицами и другими файлами в этой БД.

Главный каталог таблиц **Sysdbs** содержит данные о всех БД, созданных пользователями, включая имя БД, местоположение на диске (маршрут по директориям), имя пользователя, создавшего БД (имя пользователя в ЛВС), и дату создания БД. Файл главного каталога таблиц **Sysdbs** размещается в том же директории, что и системные файлы SQL.

4.2. КОМАНДЫ СОЗДАНИЯ БАЗ ДАННЫХ И ТАБЛИЦ SQL

В языке SQL dBASE IV имеется пять команд, связанных с созданием и использованием БД: **CREATE DATABASE, SHOW DATABASE, START DATABASE, STOP DATABASE** и **DROP DATABASE**. Создание БД осуществляется только в режиме набора с клавиатуры командой

CREATE DATABASE [<путь по директориям>] <имя БД>;

которая открывает в текущем директории или в директории, заданном маршрутом, поддиректорий с именем БД. В этом поддиректории формируются таблицы каталога, в которые помещаются данные о всех создаваемых в данном поддиректории таблицах SQL. Имя БД, пользователя, создавшего БД (в среде ЛВС), дата создания и маршрут по директориям этой БД помещаются в главный каталог **Sysdbs**.

Таким образом, БД SQL - это специально созданный директорий, в котором размещается каталог таблиц, созданных в этом директории, таблицы и другие связанные с ними файлы. Имя БД должно быть уникальным. Если при открытии БД уже существует директорий с таким же именем, то этот директорий определяется как БД SQL и в нем размещаются все компоненты БД. В каждый текущий момент может быть открыта только одна БД, в которую вводятся создаваемые таблицы. Поэтому, если требуется открыть другую БД, надо предварительно закрыть текущую.

Закрытие БД выполняется командой **STOP DATABASE**. Активизация (открытие) имеющейся БД осуществляется командой **START DATABASE [<имя БД>]**;. При этом БД, ранее открытая и не закрытая в момент активизации другой БД, автоматически закрывается. После этого можно выполнять любые команды SQL по определению таблиц, поиску данных и т.д. Для удаления БД используется команда **DROP DATABASE <имя БД>;**. Эта команда удаляет закрытую БД, а также все ее компоненты (в том числе связанные файлы **.dbf** и **.mdx**), каталоги таблиц и данные об этой БД в главном каталоге таблиц **Sysdbs**. При выполнении команды высвечивается предупреждение о выполнении операции удаления. Для подтверждения нужно дать ответ **Y** или **Yes**. Операцию можно отменить нажатием клавиши **Esc**.

Удалять можно только закрытые БД, поэтому перед удалением БД целесообразно выполнить команду ее закрытия **STOP DATABASE**. Для просмотра всех созданных БД используется команда **SHOW DATABASE**, высвечивающая список имен БД в текущем директории, имя создавшего БД пользователя (в среде ЛВС), дату создания и маршрут по директориям.

4.3. ТАБЛИЦЫ SQL

Таблицы представляют собой основной компонент БД SQL, в которых определяется структура данных. Структура таблицы задается командой **CREATE TABLE** и может быть впоследствии модифицирована (добавлены колонки) командой **ALTER TABLE**. Команды **INSERT**, **UPDATE** и **DELETE** позволяют добавлять, изменять или удалять данные из таблиц.

В dBASE IV имеются утилиты, позволяющие импортировать внешние файлы в файлы SQL (вводить и добавлять в таблицы данные из не-SQL файлов) и экспортировать файлы SQL.

Команда **DBDEFINE** дает возможность использовать в SQL файлы dBASE IV, команда **LOAD DATA** - импортировать файлы dBASE II, dBASE III и dBASE III+, данные из электронных таблиц Lotus 1-2-3, MultiPlan и Visicalc, файлы ASCII и файлы с разделителями, базы данных и электронные таблицы Framework II и RapidFile. Команда **UNLOAD DATA** экспортирует таблицы SQL в эти типы файлов.

Для создания новой таблицы SQL и определения типов данных используется команда

```
CREATE TABLE <имя таблицы>
        (<имя колонки><тип данных>
        [,< имя колонки><тип данных>...]);
```

Пример:

```
CREATE TABLE Upr1
(Tab_no CHAR(4),
Fam     CHAR(15),
Dat_rab DATE,
Adres   CHAR(30),
Oklad   Decimal(4,0),
Prem    Decimal(4,0));
```

Ниже приводятся типы данных, разрешенные для использования в таблицах SQL:

SMALLINT - целые числа до 6 значащих цифр, включая знак. Диапазон от -99,999 до 999,999;

INTEGER - целые числа до 11 значащих цифр, включая знак. Диапазон от -9,999,999,999 до 99,999,999,999;

DECIMAL(x,y) - числа с фиксированной десятичной точкой с x (от 1 до 19) позициями (включая знак числа) и y (от 0 до 18) дробными позициями справа от точки. Например, DECIMAL(6,2) позволяет вводить числа без знака до 9999.99 и числа со знаком от -999.99 до 9999.99;

NUMERIC(x,y) - числа с фиксированной десятичной точкой с x (от 1 до 20) позициями (включая знак числа и десятичную точку) и y (от 0 до 18) дробными позициями справа от точки. Например, NUMERIC(6,2) позволяет вводить числа без знака до 999.99 и числа со знаком от -99.99 до 999.99;

FLOAT(x,y) - числа с плавающей точкой с x (от 1 до 20) позициями (включая знак числа и десятичную точку) и y (от 0 до 18) дробными позициями справа от точки. Диапазон от 0.1 x 10 EXP(-307) до 0.9 x EXP(+ 308) (например, -9.99E + 235);

CHAR(n) - строка из n (до 254) символов. В колонку с этим типом данных могут вводиться данные из символьных колонок, символьных переменных и символьных строк;

DATE - дата в формате, определенном командами SET DATE и SET CENTURY (по умолчанию формат mm/dd/yy). Данные могут вводиться из колонок и переменных с этим же типом или из строк, заданных в виде { / / }, и строк, преобразованных CTOD(), например CTOD("02/15/89");

LOGICAL - логические значения .T. (истина) и .F. (ложь). Значения могут вводиться из логических переменных или колонок или в виде констант .T., .t., .Y., .y., .F., .f., .N., .n..

4.4. ВВОД, ОБНОВЛЕНИЕ И УДАЛЕНИЕ ДАННЫХ, МОДИФИКАЦИЯ ТАБЛИЦ И СОЗДАНИЕ ВЫБОРОК

Для ввода новых строк данных используются команды **INSERT** и **LOAD DATA**. Ввод конкретных значений в заданные колонки таблицы выполняется командой

INSERT INTO <имя таблицы>
[(<список колонок>)]
VALUES(<список значений>);

Пример:

```
INSERT INTO Upr1
VALUES ("0112", "Петров", {20/10/89},300);
```

Если список колонок опущен, то значения вводятся по порядку во все колонки.

Для ввода отобранных строк из существующей таблицы используется предложение **SELECT**:

INSERT INTO <имя таблицы>
[(<список колонок>)]
<предложение SELECT>;

Предложение **SELECT** представляет собой команду для извлечения данных из заданных колонок таблицы в соответствии с заданным условием

SELECT <колонки>
FROM <таблицы>
[WHERE <условие>]...;

Пример:

```
INSERT INTO Upr1
SELECT *
FROM Upr2
WHERE Tab_no> 0112
```

Символ * обозначает, что в предложении **SELECT** выбираются все колонки таблицы Upr2.

Для изменения значений строк в таблице в соответствии с заданным условием используется команда

UPDATE <имя таблицы>
SET <имя колонки> = <выражение>
[,<имя колонки> = < выражение>...]
[WHERE <условие>];

Пример:

```
UPDATE Upr1
SET Prem = Prem * 1.3
WHERE Dat_rab < CTOD("01/01/75");
```

Удаление строк таблицы по заданному условию можно выполнить командой

DELETE FROM <имя таблицы>
[WHERE <условие>];

Пример:

```
DELETE FROM Upr1
WHERE Fam = "Котов";
```

В языке SQL можно добавлять колонки в существующую таблицу, но нельзя их удалять или изменять. Для модификации таблиц целесообразно создать новую таблицу и затем ввести в нее строки из существующей таблицы.

Добавление колонок в существующую таблицу выполняется командой

ALTER TABLE <имя таблицы>
ADD (<имя колонки> <тип данных>
[,< имя колонки><тип данных>...]);

Пример:

```
ALTER TABLE Upr1
ADD (Telefon CHAR(9));
```

Для создания выборки с заданным именем и именами колонок используется команда

CREATE VIEW <имя выборки>
[(<список колонок выборки>)]
AS <предложение SELECT>
[WITH CHECK OPTION];

Пример:

```
CREATE VIEW Ord
AS SELECT Tab_no, Fam, Oklad
FROM Upr1
WHERE Adres = "Ордынка"
```

Если список колонок выборки не указан, то имена выбираются из списка в предложении **SELECT**, которое определяет строки и колонки из таблицы, включаемой в выборку.

Опция **WITH CHECK OPTION** используется для выборок, которые могут обновляться, и устанавливает, что вводимые или обновляемые строки должны проверяться на соответствие с заданными условиями. Например, эта опция позволяет запретить ввод или обновление строк выборки, которые не удовлетворяют условию, заданному в предложении **SELECT** (с ключевым словом **WHERE**).

Выборки позволяют реконструировать данные в БД, не прибегая к созданию новых таблиц или модификации старых. Как уже указывалось, выборки фактически не содержат данных и поэтому занимают значительно меньше места в памяти, чем таблицы. Можно создать несколько выборок по одной таблице, отражающих все изменения, сделанные в таблицах.

Можно создавать выборки из нескольких таблиц. Для этого в предложении **AS SELECT** указываются имена колонок из различных таблиц в виде **<имя таблицы>.<имя колонки> [,<имя таблицы> .<имя колонки>]...**

Пример:

```
CREATE VIEW Zakaz
(Zak_no, Dat_prod, Magazin)
AS SELECT Prodano.Zak_no, Prodano.Dat_prod,
Upravl.Fio
FROM Prodano, Upravl
WHERE Prodano.Upr_no = Upravl.Upr_no
```

Удаление выборки осуществляется командой

DROP VIEW <имя выборки>;

В язык SQL включены синонимы, которые замещают имя таблиц или выборок в командах. Синонимы создаются командой

CREATE SYNONYM <имя синонима>
FOR <имя таблицы или выборки>;

Синонимы используются в любых командах вместо имени таблицы или выборки с целью сокращения имени записи, например, в командах **SELECT**, **CREATE INDEX**, **INSERT**, **UPDATE**, **DELETE** и **DROP**.

Одной из важнейших характеристик любой БД является способность быстро находить нужную информацию. Индексные файлы (индексы) представляют собой одно из средств для

быстрого поиска данных в БД SQL. Индексные файлы содержат информацию о расположении строк в БД SQL в соответствии с их значениями колонок. В отличие от индексов языка dBASE IV в языке SQL не требуется указывать индекс для поиска данных. В режиме SQL автоматически выбирается тот индекс, который обеспечивает наиболее быстрый поиск. Индексы строятся только для таблиц, но не для выборок. Однако при работе с выборками обычно используются индексы, построенные для соответствующих таблиц.

Индексы для БД SQL представляют собой указатели в множественном индексном файле **.mdx**, который имеет то же имя, что и таблица SQL. Для каждой таблицы БД можно создать до 47 указателей индексов. Для создания индекса используется команда

CREATE [UNIQUE] INDEX <имя индекса>
ON <имя таблицы> (<имя колонки> [ASC/DESC]
[,<имя колонки> [ASC/DESC]...);

Указатели индексов создаются для заданных имен колонок. Имена колонок могут занимать не более 100 позиций в строке команды.

Опция **UNIQUE** определяет уникальный индекс, т.е. в индекс включаются уникальные значения колонок. При вводе или обновлении данных в таблицу с уникальным индексом записываются только уникальные значения. При создании индекса SQL таблицы связанный с ней файл БД dBASE IV можно только читать.

Опции **ASC/DESC** определяют порядок построения индекса соответственно по возрастанию (по умолчанию) или убыванию.

Индексы можно создавать по всем полям, кроме логических.

Удаление индексов осуществляется командой

DROP INDEX <имя индекса>;

Как уже указывалось выше, вся информация о компонентах БД SQL (таблицы, выборки, синонимы, индексы) хранится в системных каталогах. Каталог **Systabls** представляет собой таблицу с колонками, в которых содержится информация о существующих таблицах (**Tbname**) и их типах (**Tbtype**). Для вывода информации из этого каталога можно использовать команду

SELECT Tbname, Tbtype
FROM Systabls;

В поле **Tbtype** указывается тип таблиц данных: базовая таблица (**T**), выборка (**V**) или временная таблица (**D** или **K**). Информация о колонках таблиц хранится в каталоге **Syscols** с

колонками **Colname** (имя колонки), **Coltype** (тип колонки), **Collen** (ширина колонки), которую можно извлечь командой

```
SELECT Colname, Coltype, Collen
FROM Syscols
WHERE Tbname = <"имя таблицы">;
```

4.5. ЗАПРОСЫ НА ИЗВЛЕЧЕНИЕ ДАННЫХ В ЯЗЫКЕ SQL

Большинство запросов на извлечение данных из БД SQL строится на основе команды **SELECT**. Синтаксис этой команды позволяет создавать простые по конструкции предложения для реализации сложных запросов на извлечение, ввод и удаление данных:

```
SELECT <предложение>
[INTO <предложение>]
FROM <предложение>
[WHERE<предложение>]
[GROUP BY <предложение>]
[HAVING <предложение>]
[UNION SELECT <предложение>]...
[ORDER BY <предложение>
        /FOR UPDATE OF <предложение>]
[SAVE TO TEMP <предложение>];
```

Выше уже рассматривалась простейшая форма этой команды **SELECT <колонки> FROM <таблицы>;**.

Итогом выполнения команды **SELECT** является результирующая таблица, которую можно сохранить командой **SAVE TO TEMP**. Можно также сохранить ее в виде файла **.dbf** с помощью опции **KEEP**. В качестве заголовков колонок этой таблицы используются имена колонок, из которых извлекались данные. Если информация выбиралась из нескольких таблиц (указывется в предложении **FROM**), то перед именем колонки в результирующей таблице ставится имя таблицы, из которой выбирались данные. При выводе больших таблиц можно устанавливать команду **SET PAUSE ON**, позволяющую выводить данные страницами экрана. Кроме этого, для остановки и возобновления вывода на экран данных (при повторном нажатии) можно использовать клавиши **Ctrl-S**. Для уменьшения ширины выводимой на экран колонки целесообразно использовать в предложении **FROM** вместо ее имени алиасы или синонимы, поскольку минимальная ширина выводимой колонки определяется ее именем. Кроме того, можно вообще удалить имена колонок, используя команду **SET HEADINGS OFF**. Как уже упоминалось выше, для высвечивания всех колонок таблицы нужно в команде вместо имен колонок поместить символ (*):

SELECT *
FROM <имя таблицы>;

Для того чтобы вывести только уникальные строки таблицы, не повторяющиеся в заданных колонках, используется ключевое слово **DISTINCT**:

SELECT DISTINCT <колонки>
FROM <имя таблицы>;

Пример:

```
SELECT DISTINCT Det_no, Opisanie
FROM Detal;
```

Для отбора строк по заданному критерию используется предложение **WHERE**:

SELECT <колонки>
FROM <имя таблицы>
WHERE <условие>;

Пример:

```
SELECT Det_no, Opisanie, Sklad, Adres, Kol_det
FROM Mto
WHERE Adres = "Минск";
```

В состав критерия отбора, как и в языке dBASE IV, могут входить элементарные логические условия и сложные, в которых элементарные условия объединяются логическими связками. Элементарное логическое условие строится по форме

<имя колонки><оператор отношения><значение>

В языке SQL используются те же операторы отношений, что и в dBASE IV, за исключением оператора отрицания, который в SQL обозначается символом !.

Содержимое колонки и сравниваемого значения должны быть одного типа. Все числовые типы данных могут сравниваться между собой. Символьные константы должны заключаться в двойные или одинарные кавычки. Сравнение символьных данных между собой выполняется по их кодам ASCII, что следует учитывать при сравнении прописных и строчных букв. При формировании элементарного условия для логического типа данных можно не использовать оператор и значение. В этом случае по умолчанию предполагается оператор (=) и значение **.T.**

Пример:

```
SELECT *
FROM Zakazy
WHERE Otgruzka = .T.;
```

Последний оператор можно записать в виде **WHERE Otgruzka**. Имя в элементарном логическом условии можно использовать как аргумент в функциях, допустимых в SQL, например, в функции **UPPER()** для преобразования строчных букв в прописные. Выше приводился пример использования функции **CTOD()** для преобразования строки типа даты при сравнении с колонкой типа даты.

В качестве логических связок используются операторы **NOT** (НЕТ), **AND** (И) и **OR** (ИЛИ). Заметим, что в отличие от языка dBASE IV перед и после этих операторов не ставятся точки. Сложные логические условия оцениваются по правилам булевой алгебры. При оценке последовательности элементарных логических условий с логическими связками высший приоритет имеет оператор **NOT**, затем **AND** и **OR**.

В состав предложений **SELECT**, **WHERE** и **HAVING** могут входить выражения, включающие имена колонок, константы, переменные, математические операторы (те же, что и в dBASE IV) и функции. В предложении **SELECT** можно задавать вывод символьного выражения, а также вычисляемую колонку. Вычисляемые колонки имеют заголовки с именами EXP1, EXP2 и т.д. в соответствии с порядком их определения в предложении **SELECT**.

Пример:

```
SELECT Tab_no, Fam, "Оклад+премия", Oklad+Prem
```

В этом примере вычисляется сумма оклада и премии (Oklad + Prem), которая выводится с заголовком EXP2 и перед этой колонкой выводятся слова Оклад + премия в каждой строке с заголовком EXP1.

В условия, задаваемые в предложении **WHERE** (включая предикаты **BETWEEN**, **IN** и **LIKE**), также могут входить выражения.

Кроме констант, значений колонок и математических операторов в состав выражений могут входить многие функции, определенные в dBASE IV. Вместе с языком SQL используется их свыше 100. При этом необходимо указывать полное имя функции.

Кроме функций из языка dBASE IV в SQL входят специальные функции агрегирования, которые используются в команде **SELECT**:

COUNT() - определяет количество отобранных строк;

SUM() - суммирует значения числовых колонок;

MIN() - находит минимальное значение символьных, числовых и колонок типа даты;

MAX() - находит максимальное значение символьных, числовых и колонок типа даты;

AVG() - вычисляет среднее значение числовой колонки.

При использовании с символьными колонками функции **MIN()** и **MAX()** возвращают данные с минимальным и максимальным значениями кодов ASCII строк. Сравнение выполняется с первого символа слева. В выражениях колонки должны быть одного типа. Все числовые типы данных могут использоваться совместно. Символьные константы должны заключаться в одинарные или двойные кавычки.

Функции агрегирования могут включаться в команду **SELECT** вместо имен колонок.

Примеры:

```
SELECT COUNT(*)
FROM Upr1;
SELECT SUM(Oklad)
FROM Upr1;
SELECT MAX(Oklad), MIN(Oklad), AVG(Oklad)
FROM Upr1;
```

При выводе значений функций агрегирования сверху высвечивается наименование функции и порядковый номер, указывающий ее позицию в предложении **SELECT**. Символ * в функции **COUNT()** указывает на подсчет по любой колонке. В предложении **WHERE** могут использоваться предикаты **BETWEEN**, **IN** и **LIKE**, которые позволяют записывать условие в более компактном виде.

Предикат **BETWEEN** определяет диапазон поисковых значений. Например, вместо записи условия WHERE Oklad >= 300 AND Oklad <= 500 можно записать WHERE Oklad BETWEEN 300 AND 500. Этот предикат указывает закрытый интервал значений.

Предикат **IN** проверяет, находится ли значение колонки среди указанных в списке. Например, вместо записи условия WHERE sokr = "АБ" OR "ВГ" OR "ДЕ" можно записать WHERE sokr IN ("АБ", "ВГ", "ДЕ").

Предикат **LIKE** осуществляет поиск на включение подстроки, задаваемой в переменной или в виде константы. Подстрока определяется заданными символами, окруженными символами замещения (_) и (%). Первый из них замещает один символ, второй любое число символов.

Пример:

```
SELECT Tab_no, Fam, Oklad
FROM Upr1
WHERE Fam LIKE "Пет%";
```

Как и обычные условия, в предложении **WHERE** предикаты **BETWEEN**, **IN** и **LIKE** могут объединяться логическими связками **AND**, **OR** и **NOT**. При объединении допускаются скобки для облегчения построения сложных условий.

При отборе строк из таблицы они обычно выводятся в том порядке, в котором создана таблица. Для изменения порядка выводимых строк используется предложение **ORDER BY**:

SELECT <колонки>
FROM <таблицы>
WHERE <условие>
ORDER BY <колонка/целое число> [ASC/DESC]
[,< колонка/целое число> [ASC/DESC]...];

Колонки, используемые в предложении **ORDER BY**, должны быть включены в предложение **SELECT**. В предложении **ORDER BY** вместо имен колонок можно ставить целые числа, определяющие их позицию в предложении **SELECT**. Для ускорения извлечения данных целесообразно командой **CREATE INDEX** создавать индексы по поисковым колонкам. Ключевые слова **ASC/DESC** указывают на упорядочение по возрастанию или убыванию.

При упорядочивании таблицы более чем по одной колонке строки располагаются в порядке, определяемом первой колонкой. Строки с одинаковым значением этой колонки располагаются в порядке, определяемом второй колонкой, и т.д.

Два дополнительных предложения **GROUP BY** и **HAVING** позволяют располагать строки по группам. Затем можно выполнять операции с этими группами, например используя функции агрегирования. Предложение **GROUP BY** позволяет объединить строки результирующей таблицы, определяемой предложением **SELECT**, в группы с одинаковым значением заданных колонок. После агрегирования каждая такая группа сводится к одной строке в результирующей таблице. В качестве имени колонки с результатом агрегирования используется имя функции агрегирования с порядковым номером этой функции.

Пример:

```
SELECT Det_no, Opisanie, Kol_det
FROM Mto
GROUP BY Det_no, Opisanie;
```

Существуют два ограничения при использовании предложения **GROUP BY**. Во-первых, все колонки, указанные в предложении **SELECT**, должны быть включены в предложение **GROUP BY** и наоборот. Во-вторых, в предложении **GROUP BY** нельзя указывать вычисляемую колонку за исключением имен колонок, использованных в функциях агрегирования в предложении **SELECT**.

Пример:

```
SELECT Oblast, COUNT(*)
FROM Naselenie
GROUP BY Oblast;
```

В этом примере подсчитывается количество жителей в областях.

Предложение **GROUP BY** может использоваться совместно с предложением **ORDER BY** для обеспечения упорядоченного расположения выводимых строк по заданной колонке.

Пример:

```
SELECT Det_no, Opisanie, Sum(Kol_det)
FROM Sklad
GROUP BY Opisanie,Det_no
ORDER BY Opisanie;
```

Предложение **HAVING**, используемое вместе с предложением **GROUP BY**, позволяет отобрать в состав групп строки, удовлетворяющие заданным условиям. Действие предложения **HAVING** аналогично предложению **WHERE**, но определяет условие, которому должна удовлетворять каждая группа для вывода в результирующую таблицу. Например, для вывода упорядоченного по алфавиту перечня деталей стоимостью более 300 руб., которых на складе больше 10, можно использовать следующую команду **SELECT**:

```
SELECT Opisanie, Nomer, SUM (Kolichestvo), Stoimost
FROM Sklad
WHERE Stoimost > 300
GROUP BY Opisanie, Nomer, Stoimost
HAVING SUM (Kolichestvo) > 10
ORDER BY Opisanie;
```

Если предложение **HAVING** используется без предложения **GROUP BY**, то его действие распространяется на всю результирующую таблицу и эквивалентно предложению **WHERE**.

В языке SQL имеется еще один способ реализации сложных запросов - с помощью команды **UNION**. Эта команда позволяет объединить результирующие таблицы, получаемые в результате выполнения нескольких команд **SELECT**, удаляя при этом все дублируемые строки и оставляя из них только одну.

При этом результирующие таблицы из всех запросов должны быть совместимы, т.е. должны иметь одинаковое количество колонок и совместимые типы данных в соответствующих колонках. Имена колонок в каждом запросе могут быть различными.

Для символьных колонок их ширина должна быть одинаковой. Для числовых колонок должно быть определено одинаковое общее количество цифр, одинаковый тип числа (с фиксированной или плавающей точкой) и одинаковое число дробных знаков.

Для упорядочивания результирующей таблицы после выполнения команд **UNION** нужно включить предложение **ORDER BY** в последнюю команду **SELECT**. При этом в предложении **ORDER BY** для указания колонки, по которой упорядочивается результи-

рующая таблица, необходимо использовать число, определяющее позицию колонки в командах **SELECT**. Например, для выбора всех городов, в которых расположены заводы, поставщики и склады производственного объединения, можно использовать следующие команды:

```
SELECT Gorod
FROM Zavod
    UNION
SELECT Mesto
FROM Postavchik
    UNION
SELECT Mesto
FROM Sklad
    ORDER BY 1;
```

4.6. ИЗВЛЕЧЕНИЕ ДАННЫХ ИЗ НЕСКОЛЬКИХ ТАБЛИЦ И ВЛОЖЕННЫЕ ЗАПРОСЫ

Команда **SELECT** позволяет выполнять запросы на выборку данных не только из одной таблицы, но и из нескольких таблиц. При этом используются те же самые предложения, что и при запросах из одной таблицы. В предложении **FROM** через запятую указываются имена таблиц. Если в нескольких таблицах имеются колонки с одинаковым именем, то тогда перед этими именами указываются имена таблиц, отделенные точкой. В случае, если требуется вывести все колонки какой-либо таблицы, можно вместо имен колонок в составном имени указать символ замещения *, например: Sklad.* (из таблицы с именем Sklad выводятся все колонки).

Пример:

```
SELECT Sklad1.*, (Kolichestvo * Stoimost)
FROM Sklad1, Sklad2
WHERE Sklad1.Detal = Sklad2.Detal AND Kolichestvo > 10
```

В этом примере выводятся все колонки таблицы Sklad1 (склад N1 готовых деталей) и, кроме того, вычисляемая колонка - произведение количества имеющихся деталей на их стоимость. При этом отбираются те строки таблицы Sklad1, в которых совпадают значения колонки Detal (наименование детали) с аналогичной колонкой в таблице Sklad2 (склад N2 готовых деталей) и их количество превышает 10 на обоих складах. В предложении **WHERE** можно использовать функции dBASE IV, например, функцию **UPPER()**, если названия деталей вводились как со строчной, так и с прописной буквы.

В результирующей таблице данных, извлеченных из нескольких таблиц, выводятся составные имена колонок, которые

бывают весьма длинными. Напомним, что для уменьшения ширины выводимой колонки можно вместо имен колонок указывать их алиасы и синонимы. Кроме того, можно вообще удалить имена колонок командой **SET HEADING OFF**. Колонки, представляющие собой функции агрегирования, имеют соответствующие заголовки: **AVG, COUNT, MAX** и **SUM** с цифрой справа, определяющей позицию функции агрегирования в предложении **SELECT**.

Заголовки колонок, в которых используются функции dBASE IV, имеют имена соответствующих функций. Колонки, определенные в предложении **GROUP BY**, выводятся с именами колонок, которым предшествует **G_**.

Пример:

```
SELECT Detal.Nomer, Opisanie, COUNT(*)
FROM Detal, Prodano
WHERE Detal.Nomer=Prodano.Nomer
GROUP BY Detal.Nomer, Opisanie
ORDER BY Opisanie;
```

В этом примере выводятся строки таблицы Detal с колонками G_Nomer (номер детали), G_Opisanie (наименование детали) и COUNT1 (количество проданных деталей), в которых количество проданных деталей определяется их подсчетом из таблицы Prodano с помощью функции агрегирования **COUNT** по группам проданных номеров деталей. Результирующая таблица упорядочивается по наименованиям деталей.

Для этого же примера можно показать использование предложения **HAVING**, например, вывести только те детали, которых продано больше 10. Тогда следует добавить предложение HAVING COUNT(*) > 10.

В следующем примере показывается выполнение запроса из трех таблиц: Detal, Prodano и Komu (кому продано). Из последней таблицы выводятся наименования организаций, купивших детали (колонка Pokupatel), упорядоченные по алфавиту.

Пример:

```
SELECT  Detal.Nomer, Opisanie, Pokupatel
FROM  Detal, Prodano, Komu.
WHERE Detal.Nomer=Prodano.Nomer
AND Prodano.Pokupatel=Komu.Pokupatel
ORDER BY Komu.Pokupatel
```

Использование алиасов (альтернативных имен) таблиц позволяет выполнять запросы, в которых в предложении **WHERE** в левой и правой части условия можно ставить колонки одной таблицы. Это дает возможность осуществлять поиск в таблице по условию, включающему значения из нескольких колонок этой таблицы.

Например, имеется таблица Kadry, в которой размещены данные о кадрах предприятия с колонками Nomer (персональный номер по учету кадров), Fam (фамилия работника) и Kurator (персональный номер наставника, который курирует работника). Требуется вывести список всех работников предприятия и их кураторов. Для этого можно использовать команду

```
SELECT Rabotnik.Fam, Nastavnik.Fam
FROM Kadry Rabotnik, Kadry Nastavnik
WHERE Nastavnik.Kurator = Rabotnik.Nomer
ORDER BY Rabotnik.Fam;
```

Если требуется в первой колонке вывести кураторов, во второй слово "курирует", а в третьей работников и упорядочить кураторов по алфавиту, то команда будет выглядеть так:

```
SELECT Nastavnik.Fam, "Курирует", Rabotnik.Fam
FROM Kadry Rabotnik, Kadry Nastavnik
WHERE Nastavnik.Kurator = Rabotnik.Nomer
ORDER BY Nastavnik.Fam;
```

В состав условий в предложении **WHERE** (или **HAVING**) могут входить команды **SELECT**, образуя вложенные запросы. Команда **SELECT** в предложении **WHERE** представляет собой внутренний запрос. Данные, полученные после выполнения внутреннего запроса, затем используются для выполнения внешнего запроса.

Конструкция внешнего запроса зависит от количества значений, возвращаемых внутренним запросом. Могут быть два случая: внутренние запросы возвращают одно значение или несколько значений (более одной строки значений одной колонки).

Существует два основных типа вложенных запросов: простые и коррелированные. В простых вложенных запросах внутренний запрос выполняется независимо от внешнего запроса и его результаты используются для выполнения внешнего запроса. В коррелированном вложенном запросе внутренний запрос обращается одновременно к своей таблице данных и таблице данных внешнего запроса и его результаты используются для выполнения внешнего запроса.

В качестве примера для иллюстрации вложенного запроса, в котором внутренний запрос возвращает одно значение, рассмотрим таблицу Prodano с колонками Detal (наименование проданной детали), Data (дата продажи), Indpokup (индекс покупателя) и таблицу Komu с колонками Pokupatel (наименование организации - покупателя) и Indpokup (индекс покупателя). Для вывода наименования деталей и даты их продажи заводу "Уралмаш" можно выполнить запрос:

```
SELECT Detal, Data
FROM Prodano
WHERE Indpokup =
```

```
   (SELECT Indpokup
   FROM Komu
   WHERE Pokupatel = "Уралмаш")
ORDER BY Detal;
```

Для возврата единственного значения из внутреннего запроса часто используются функции агрегирования:

```
SELECT Fam, Oklad
FROM Kadry
WHERE Oklad >
   (SELECT AVG(Oklad)
    FROM Kadry)
ORDER BY Fam;
```

В последнем примере из таблицы Kadry (персональный учет кадров) с колонками Fam (фамилия) и Oklad (зарплата) выбираются работники, получающие зарплату выше средней.

В предложении **WHERE** можно включать несколько внутренних запросов, объединенных логическими операторами **AND**, **OR** и **NOT** для построения сложных условий.

Пример:

```
SELECT Detal, Data
FROM Prodano
WHERE Indpokup =
(SELECT Indpokup
FROM Komu
WHERE Pokupatel = "Уралмаш")
OR Zavodnomer =
(SELECT Zavodnomer
FROM Proizvod
WHERE Nazvanie = "Знамя труда")
ORDER BY Detal;
```

В данном примере выводятся детали, проданные "Уралмашу" или произведенные заводом "Знамя труда".

Если внутренний запрос возвращает не одну, а несколько строк, то в предложении **WHERE** необходимо включать опции, позволяющие воспринять несколько значений, например, ключевые слова **IN**, **ANY** и **ALL**.

Пример:

```
SELECT Detal, Data
FROM Prodano
WHERE Indpokup IN
(SELECT Indpokup
FROM Komu
WHERE Pochtind LIKE "7%")
ORDER BY Detal;
```

В последнем примере внутренний запрос отбирает организации-покупатели, у которых почтовый индекс (колонка Pochtind)

начинается с 7, затем внешний запрос в соответствии с индексом отобранных организаций (колонка Indpokup) выводит купленные ими детали. Следует отметить, что во внутреннем запросе нельзя использовать предложения **ORDER BY** и **UNION**. Внутренние запросы могут содержать предложения **GROUP BY** и **HAVING**, но в этом случае нельзя их использовать повторно в целом запросе.

Проиллюстрируем использование ключевых слов **ANY** и **ALL**.

Пример:

```
SELECT Detal, Kolich, Mesto
FROM Sklady
WHERE Kolich < ANY
(SELECT Kolich
FROM Sklady
WHERE Mesto = "Минск")
ORDER BY Detal;
```

В приведенном примере выводятся наименования деталей (колонка Detal) и их количество (колонка Kolich) на тех складах (таблица Sklady), на которых деталей меньше, чем на любом другом складе в Минске.

Пример:

```
SELECT Detal, Kolich, Mesto
FROM Sklady
WHERE Kolich > ALL
(SELECT Kolich
FROM Sklady
WHERE Mesto = "Киев")
ORDER BY Detal;
```

В данном примере выводятся наименования деталей и их количество на тех складах, на которых деталей больше, чем на других складах в Киеве.

До сих пор рассматривались вложенные запросы с двумя уровнями вложенности. В принципе можно создавать запросы с неограниченным количеством уровней.

Пример:

```
SELECT Indorg, Organiz
FROM Komu
WHERE Indorg IN
(SELECT Indorg
FROM Prodano
WHERE Nomzakaz IN
(SELECT Nomzakaz
FROM Zakaz
WHERE Detnomer = "005679"))
ORDER BY Organiz;
```

В последнем примере показан вложенный трехуровневый запрос, с помощью которого выводятся организации, которые купили (таблица Komu) детали с заданным индексом. Запрос нижнего уровня возвращает из таблицы Zakaz номера заказов (колонка Nomzakaz), в которые входят детали с номером 005679. Запрос следующего вверх уровня возвращает из таблицы Prodano индексы организаций, в заказы которых (колонка Nomzakaz) входили детали с номером 005679. И, наконец, запрос самого верхнего уровня возвращает из таблицы Komu наименования (колонка Organiz) и индексы организаций (колонка Indorg), купивших детали с номером 005679.

Этот же запрос можно записать в виде одноуровневой структуры, используя поиск по нескольким таблицам со сложным условием, заданным в предложении **WHERE**:

```
SELECT Indorg, Organiz
FROM Komu, Prodano, Zakaz
WHERE Indorg = Prodano.Nomzakaz
   AND Zakaz.Nomzakaz = Zakaz.Detnomer
   AND Zakaz.Detnomer = "005679"
ORDER BY Organiz;
```

Вообще любые вложенные запросы можно представить в виде одноуровнего запроса по нескольким таблицам (обратное не верно: не все одноуровневые запросы по нескольким таблицам можно представить в виде вложенных структур). Эти формы эквивалентны по своему результату, и пользователи могут выбирать ту структуру, которая кажется им нагляднее для записи запроса.

В предложении **WHERE** может использоваться предикат **EXISTS**, позволяющий сформировать условие, которое оценивает результат выполнения вложенного запроса в целом, в отличие от оценки отдельных значений, возвращаемых внутренним запросом. Этот предикат возвращает логические значения "Истина" или "Ложь" в зависимости от того, возвращает или нет внутренний запрос какое-либо значение. Предикат **NOT EXISTS** возвращает соответственно инвертированные (обратные) логические значения.

Пример:

```
SELECT Indorg, Organiz, Mesto
FROM Komu
WHERE Exists
(SELECT *
FROM Prodano
WHERE Komu.Indorg = Prodano.Indorg)
ORDER BY Organiz;
```

В этом примере из таблицы Komu выводятся те организации, которым что-либо продано, определяемое условием в

предложении **WHERE**. Можно, наоборот, вывести организации, которым ничего не продано, используя вместо предиката **EXISTS** предикат **NOT EXISTS**.

Предикаты **EXISTS** и **NOT EXISTS** можно использовать многократно в многоуровневых вложенных запросах. Следует отметить, что только для этих предикатов во вложенных запросах разрешается в предложении **SELECT** использовать знак вывода всех колонок *.

Наиболее сложной формой вложенных запросов является коррелированный запрос. В этом типе запроса данные из таблицы внешнего запроса используются при выполнении внутреннего запроса.

Если во внутреннем и внешнем запросах используется одна и та же таблица, то необходимо задавать алиас (псевдоним) таблицы, так как формально могут сравниваться только различные таблицы.

Пример:

```
SELECT Fam, Otdel
FROM Kadry
WHERE Oklad =
(SELECT MAX(Oklad)
FROM Kadry Synonim
WHERE Synonim.Otdel = Kadry.Otdel)
ORDER BY Fam;
```

В приведенном примере во внутреннем запросе используется таблица из внешнего запроса, являющаяся одной и той же таблицей персонального учета кадров. Во внутреннем запросе определяется максимальный оклад сотрудников в отделах. Во внешнем запросе выполняется сравнение окладов сотрудников каждого отдела с максимальным окладом в соответствующем отделе. Если оклад равен максимальному, то эта строка выводится в результирующую таблицу.

В следующем примере из таблицы Prodano выводятся все индексы организаций (колонка Indpokup), сделавших более одного заказа (колонка Nomzakaz).

Пример:

```
SELECT Indpokup, Nomzakaz
FROM Prodano
WHERE Indpokup IN
(SELECT Synprod.Indpokup
FROM Prodano Synprod
WHERE Nomzakaz <> Synprod.Nomzakaz)
ORDER BY Indpokup, Nomzakaz;
```

4.7. СОВМЕСТНОЕ ИСПОЛЬЗОВАНИЕ КОМАНД SQL И dBASE

Как уже указывалось, в СУБД dBASE IV можно совместно использовать команды SQL и команды и функции языка dBASE IV с целью выполнения различных операций при работе с БД. Пользуясь только командами SQL, можно создавать и удалять таблицы данных и их индексы, вводить, обновлять, удалять или отбирать данные. Кроме того, можно устанавливать привилегии для доступа к данным. Функции агрегирования SQL позволяют выполнять простые статистические операции **(SUM(), COUNT(), AVG(), MIN()** и **MAX())**.

Язык dBASE IV можно применять в командном режиме для создания и распечатки отчетов с данными, отобранными запросами на языке SQL.

Команды и функции языка dBASE IV можно также использовать для преобразования или форматирования данных, полученных по запросам на языке SQL, установки параметров и состояний среды и разработки основной части прикладных программ.

Напомним, что для выполнения команд SQL и dBASE IV предусмотрены раздельные режимы из-за различия систем доступа к данным. При наборе и выполнении команд с клавиатуры можно переходить из одного режима в другой, выполняя команды SQL или dBASE IV по обработке одних и тех же данных. При этом язык SQL целесообразно использовать для отбора данных, а язык dBASE для организации форматированного ввода и вывода.

Команды языка dBASE IV, разрешенные в режиме SQL

!
?, ?? или ???
&&
*
@...CLEAR
@...DOUBLE/PANEL/NONE...
@...FILL TO...
@...SAY...GET
ACCEPT
ACTIVATE MENU
ACTIVATE POPUP
ACTIVATE SCREEN
ACTIVATE WINDOW
BEGIN...END TRANSACTION
CALL
CANCEL
CLEAR GETS
CLEAR MENUS
CLEAR POPUPS
CLEAR TYPEAHEAD
CLEAR WINDOWS
CLOSE ALTERNATE
CLOSE FORMAT
CLOSE PROCEDURE
COMPILE
COPY FILE
DEACTIVATE MENU
DEACTIVATE POPUP
DEACTIVATE WINDOW
DEBUG
DECLARE
DEFINE BAR
DEFINE BOX
DEFINE MENU

DEFINE PAD
DEFINE POPUP
DEFINE WINDOW
DELETE FILE
DIR/DIRECTORY
DISPLAY FILES
DISPLAY HISTORY
DISPLAY MEMORY
DISPLAY STATUS
DISPLAY USERS
DO
DO CASE...CASE
OTHERWISE...ENDCASE
DO WHILE...ENDDO
EJECT
ERASE
EXIT
FUNCTION
HELP
IF...ELSE...ENDIF
INPUT
LIST FILES
LIST HISTORY
LIST MEMORY
LIST STATUS
LIST USERS
LOAD
LOGOUT
LOOP
MODIFY COMMAND/FILE
MOVE GETS
MOVE WINDOW
NOTE
ON ERROR/ESCAPE/KEY
ON PAD
ON PAGE
ON READERROR
ON SELECTION PAD
ON SELECTION POPUP
PARAMETERS
PLAY MACRO
PRINTJOB...ENDPRINT
PRIVATE
PROCEDURE
PROTECT
PUBLIC
QUIT
READ [SAVE]
RELEASE
RELEASE MENUS
RELEASE MODULE
RELEASE POPUPS
RELEASE WINDOWS
RENAME
RESTORE FROM
RESTORE MACROS
RESTORE MENU
RESTORE WINDOW
RESUME
RETRY
RETURN
RUN
SAVE MACROS
SAVE TO
SAVE WINDOW
SET
SET ALTERNATE ON/OFF
SET AUTOSAVE ON/OFF
SET BELL TO
SET BELL ON/OFF
SET BORDER
SET CENTURY ON/OFF
SET CLOCK ON/OFF
SET CLOCK TO
SET COLOR OF
SET COLOR ON/OFF
SET COLOR TO
SET CONFIRM ON/OFF
SET CONSOLE ON/OFF
SET CURRENCY LEFT/RIGHT
SET CURRENCY TO
SET DATE
SET DEBUG ON/OFF
SET DECIMALS TO
SET DEFAULT TO
SET DELETED ON/OFF
SET DELIMITERS ON/OFF
SET DELIMITERS TO
SET DEVELOPMENT ON/OFF
SET DEVICE TO
SET DISPLAY TO
SET DOHISTORY ON/OFF
SET ECHO ON/OFF
SET ENCRYPTION ON/OFF

SET ESCAPE ON/OFF
SET EXACT ON/OFF
SET EXCLUSIVE ON/OFF
SET FIXED ON/OFF
SET FORMAT TO
SET FUNCTION
SET HEADING ON/OFF
SET HELP ON/OFF
SET HISTORY ON/OFF
SET HISTORY TO
SET HOURS TO
SET INTENSITY
SET LOCK ON/OFF
SET MARGIN TO
SET MARK TO
SET MENUS ON/OFF
SET MESSAGE TO
SET ODOMETER TO
SET PATH TO
SET PAUSE ON/OFF
SET POINT TO
SET PRECISION TO
SET PRINTER ON/OFF
SET PRINTER TO
SET PROCEDURE TO
SET REFRESH TO
SET REPROCESS TO
SET SAFETY ON/OFF
SET SCOREBOARD ON/OFF
SET SEPARATOR TO
SET SPACE ON/OFF
SET SQL ON/OFF
SET STATUS ON/OFF
SET STEP ON/OFF
SET TALK ON/OFF
SET TRAP ON/OFF
SET TYPEAHEAD TO
SHOW MENU
SHOW POPUP
STORE
SUSPEND
TEXT...ENDTEXT
TYPE
WAIT

Функции dBASE IV, разрешенные в режиме SQL

&*
ABS()
ACOS()
ASC()
ASIN()
AT()
ATAN()
ATN2()
BAR()*
CALL()*
CDOW()
CEILING()
CHR()
CMONTH()
COL()*
COMPLETED()*
COS()
DATE()
DAY()
DIFFERENCE()
DISKSPACE()*
DMY()
DOW()
DTOC()
DTOR()
DTOS()
ERROR()*
EXP
FILE()*
FIXED()
FKLABEL()*
FKMAX()*
FLOAT()
FLOOR()
GETENV()*
IFF()*
INKEY()*
INT()
ISALPHA()*
ISCOLOR()*
ISLOWER()*
ISUPPER()*
LASTKEY()*
LEFT()
LEN()
LIKE()*
LINENO()*
LOG()
LOG10()
LOWER()
LTRIM()
MAX()*
MDY()
MEMORY()*
MENU()*
MESSAGE()*
MIN()*
MODE()
MONTH()
NETWORK()*
OS()*
PAD()*
PCOL()*

PI()	ROW()*	SUM()
POPUP()*	RTOD()	TAN()
PRINTSTATUS()*	RTRIM()	TIME()
PROGRAM()*	SET()*	TRANSFORM()
PROMPT()*	SIGN()	TRIM()
PROW()*	SIN()	TYPE()*
RAND()	SOUNDEX()	UPPER()
READKEY()*	SPACE()	USER()
REPLICATE()	SQRT()	VAL()
RIGHT()	STR()	VARREAD()*
ROLLBACK()*	STUFF()	VERSION()*
ROUND()	SUBSTR()	YEAR()

Символом * отмечены функции, которые разрешены в режиме SQL, но не разрешены в командах SQL.

Команды и функции, которые открывают файлы БД (**USE**), устанавливают указатель записи (**RECNO()**, **GOTO** и **SKIP**) и осуществляют поиск и вывод данных (**FIND**, **SEEK** и **DISPLAY**) нельзя использовать в режиме SQL. Эти действия должны выполняться командами SQL.

В режиме SQL команды dBASE IV набираются как обычно и в отличие от команд SQL не завершаются точкой с запятой. Выполнение осуществляется нажатием клавиши **Enter**. Например, для изменения формата даты набирается команда **SET DATE TO GERMAN**.

Функции, перечисленные выше, могут использоваться в режиме SQL в командах как dBASE IV, так и SQL (не отмеченные знаком *).

Выше в примерах приводилось использование функций dBASE в предложениях **SELECT** и **WHERE** при формировании запросов. Приведем еще пример выполнения команды SQL с функциями dBASE при наборе их с клавиатуры. Например, требуется извлечь из таблицы Kadry фамилию и инициалы (колонки Fam и Init) лиц, принятых на работу до июня месяца текущего года (колонка даты приема Data1):

```
SELECT RTRIM(Fam) + SPACE(1) + Init, Data1
FROM Kadry
WHERE YEAR(Data1) = 89 AND MONTH(Data1) < 6;
```

В языке SQL пакета dBASE IV предусмотрена команда для сохранения на диске результатов выполнения запроса:

SAVE TO TEMP <имя таблицы> [(список колонок)][KEEP];

где **<имя таблицы>** - имя таблицы, в которой сохраняются результаты выполнения запроса; **(список колонок)** - список имен выводимых колонок.

Опция **KEEP** позволяет сохранить таблицу в виде файла БД dBASE IV (с расширением **.dbf**).

Эта команда полезна при разбивке сложных команд **SELECT** на небольшие, более простые операции. Разделяя такие операции, как вложенный запрос или запрос на извлечение данных из нескольких таблиц на части, можно выполнять их по этапам, проверяя результаты выполнения промежуточных операций.

Другое применение этой команды заключается в преобразовании результирующей таблицы в файл БД dBASE IV (при использовании опции **KEEP**). Затем этот файл можно использовать для вывода отчетов или марок, разработанных в УЦ dBASE IV, или продолжить обработку данных с помощью команд dBASE IV.

Если опция **KEEP** не задана, то результирующая таблица сохраняется временно, до тех пор, пока остается активным текущий файл БД, в течение текущего сеанса режима SQL или в течение выполнения процедуры высшего уровня, вызвавшей программный файл **.prs**. Можно также создать файл выборки с помощью команды **CREATE VIEW**, используя команду **SELECT** для определения строк и столбцов выборки. В случае задания опции **KEEP** для создания файла БД, можно на основе этого файла, используя команду **DBDEFINE**, создать таблицу SQL.

Пример:

```
SELECT DISTINCT Nomzakaz, Zakaz.Nomdet, Opisanie,
Kolich, Stoimed, (Kolich*Stoimed)
FROM Zakaz, Sklad
WHERE Sklad.Nomdet = Zakaz.Nomdet
ORDER BY Zakaz.Nomzakaz
SAVE TO TEMP Nalichie (Nomzakaz, Nomdet, Opisanie,
Kolich, Stoimed, Vsego);
```

Этот запрос по двум таблицам Zakaz (заказы на покупку деталей) и Sklad (наличие деталей на складе) возвращает таблицу, сохраняемую временно под именем Nalichie, в которой перечисляются заказанные детали, имеющиеся на складе. Все колонки этой таблицы те же самые, что и в результирующей таблице, заданной предложением **SELECT**, за исключением вычисляемой колонки, которая названа Vsego. Для проверки правильности выполнения запроса можно выполнить команду

```
SELECT *
FROM Nalichie;
```

Для формирования таблицы данных, необходимых при выписке накладных на покупку деталей, выполняются следующие команды:

```
SELECT DISTINCT Fam, Data, Nalichie.Nomzakaz, Nomdet,
Opisanie, Kolich, Stoimed, Vsego
```

```
FROM Pokupatel, Prodano, Nalichie
WHERE Prodano.Nomzakaz = Pokupatel.Nompokup
AND Nalichie.Nomzakaz = Prodano.Nomzakaz
ORDER BY Prodano.Nomzakaz
SAVE TO TEMP Naklad (Fam, Nomsklad, Nomzakaz, Nomdet,
Opisanie, Kolich, Stoimed, Vsego) KEEP;
```

Запрос, сформированный из приведенных команд, завершает операции по сбору всей необходимой информации для печати накладных. Результирующая таблица записывается в виде файла БД с именем Naklad. Для проверки формирования данных можно выполнить команду

```
SELECT *
FROM Naklad;
```

После этого остается создать форму (отчет) накладных и отпечатать их. Для этого можно использовать генератор отчетов УЦ dBASE IV.

Файлы БД и индексы dBASE IV можно использовать в режиме SQL, но предварительно требуется определить эти файлы как таблицы SQL. Для этого необходимо обновить каталоги таблиц SQL с помощью сервисной команды **DBDEFINE**, чтобы включить в них информацию о файлах БД и связанных индексных файлах. После того как файл БД определен как таблица SQL, его структуру уже нельзя модифицировать с помощью команд dBASE IV. В режиме dBASE можно использовать все команды dBASE IV для доступа к таблицам SQL, так как они всегда записываются с расширением **.dbf**. Можно добавлять, обновлять или удалять данные в этих файлах БД, используя команды dBASE IV со следующими ограничениями:

1. Если установлена защита данных командой **PROTECT**, то нельзя в режиме dBASE получить доступ к файлам БД, созданным в режиме SQL (можно использовать команды **SET ENCRYPTION** и **COPY** для создания дешифрованных копий этих файлов).

2. Нельзя использовать команды **MODIFY STRUCTURE** и **CONVERT TO** для файлов БД, созданных в режиме SQL.

3. При создании новых файлов БД нельзя использовать имена уже имеющихся файлов, созданных в режиме SQL.

4. Указатели связанного множественного индексного файла (имеющего то же имя, что и файл БД) нельзя удалять или добавлять, если файл БД определен как таблица SQL (с помощью команды **CREATE TABLE** или **DBDEFINE**).

5. Не разрешается обновлять созданные в SQL файлы БД, для которых в SQL установлен индекс с опцией **UNIQUE** (с помощью команды **CREATE INDEX**). Нельзя также обновлять каталоги таблиц SQL.

Упоминавшаяся выше команда **DBDEFINE [<имя .dbf файла>];** позволяет определить файл БД dBASE IV как таблицу

SQL. Эта команда формирует и вводит необходимую информацию в каталог БД SQL для обеспечения входа в файл БД и связанные с ними множественные индексные файлы, созданные в режиме dBASE, что позволяет осуществлять доступ к ним с помощью команд SQL. Команда вводит в каталог информацию об одном или всех файлах БД текущего директория.

Для того, чтобы ввести все файлы БД текущего директория в состав новой БД SQL, необходимо выполнить следующее:

1. Создать БД SQL, используя команду **CREATE DATABASE** (если требуется добавить файлы БД в существующую БД SQL, то этот пункт пропускается).

2. Скопировать файлы БД, поля **Memo** и индексные файлы в директорий, в котором требуется их хранить (для этого можно использовать команду **COPY FILE** в режиме SQL).

3. Активизировать БД, используя команду **START DATABASE**.

4. Выполнить команду **DBDEFINE**. Все файлы БД, скопированные в директорий, будут введены в таблицы каталога БД (для ввода конкретных файлов БД требуется указать их имена непосредственно в команде **DBDEFINE**).

Команда **DBDEFINE** определяет таблицы только для незашифрованных файлов БД. Для дешифрования зашифрованных в режиме dBASE файлов перед выполнением команды **DBDEFINE** следует выполнить команду **SET ENCRYPTION OFF** и затем скопировать их командой **COPY TO** для создания незашифрованных копий файлов.

При определении файлов БД как таблиц с помощью команды **DBDEFINE** нельзя включать в каталог SQL выборки файлов БД dBASE IV. Для этого сначала надо включить в каталог файлы БД, представленные в выборке, а затем создать выборку SQL, эквивалентную выборке dBASE. Кроме того, нельзя включать в каталог индексные файлы с расширением **.ndx**. Необходимо преобразовать эти файлы **.ndx** в указатели множественных индексных файлов **.mdx** и затем выполнить команду **DBDEFINE**.

После обновления каталога SQL команда **DBDEFINE** записывает текстовый файл **Dbdefine.txt** в директорий текущей БД. Этот файл содержит информацию о всех файлах БД, определенных как таблицы SQL, а также о файлах БД, для которых операция определения как таблиц SQL завершилась неуспешно.

Если при выполнении команды **DBDEFINE** включаются файлы **.mdx**, то в каталог вводятся данные о последней выполненной команде **REINDEX**. Необходимо выполнять эту команду перед установкой режима SQL, иначе данные о переупорядочивании индексного файла не будут внесены в каталог. В режиме SQL эту операцию следует выполнять командой **RUNSTATS**.

Выполнение команды **DBDEFINE** можно показать на следующем примере. Пусть требуется в командном режиме определить как таблицу SQL файл БД Kadry в текущей БД SQL:

```
DBDEFINE Kadry;
```

Для просмотра информации в таблице каталога SQL следует выполнить команду

```
SELECT *
FROM Systabls;
```

В таблице **Systabls** каждая строка содержит данные об одной таблице SQL в текущей БД. Для просмотра информации о колонках, которая введена в каталог, следует выполнить команду

```
SELECT Colno, Colname, Coltype, Collen
FROM Syscols
WHERE Tbname = "Kadry";
```

Выводимая этой командой информация позволяет убедиться в том, что в каталог введена та же структура файла БД, которая определена в dBASE.

Для проверки введенной информации о файлах БД и связанных с ними множественных индексных файлах в каталоги SQL при определении файлов БД в виде таблиц SQL используется команда

DBCHECK [<имя таблицы>];

Проверка осуществляется посредством сравнения данных в таблицах каталога SQL с действительной структурой файлов БД в текущем директории. Если нет совпадения, то выводится сообщение об ошибке. В этом случае необходимо выполнить следующее:

скопировать файлы БД и индексы **.mdx**, перечисленные в сообщении об ошибке в другой директорий;

уничтожить эти таблицы из БД SQL командой **DROP**;

скопировать файлы БД и индексы **.mdx** обратно в директорий БД;

выполнить команду **DBDEFINE <имя файла БД>** для переопределения файлов БД как таблиц SQL.

В режиме SQL имеется две команды **LOAD DATA** и **UNLOAD DATA**, которые позволяют передавать данные между БД SQL и внешними файлами данных. Можно передавать те же типы файлов, что и командами **APPEND FROM** и **COPY TO** из режима dBASE:

dBASE II, dBASE III, dBASE III+ и dBASE IV;
RapidFile (RPD);
Framework II - файлы БД и электронные таблицы (FW2);
Файлы ASCII с разделителями (DELIMITED);
Файлы ASCII системного формата данных (SDF);
VisiCalc (DIF);
Multiplan - электронные таблицы - (SYLK);
Lotus 1-2-3 (WKS).

Команды **LOAD** и **UNLOAD** могут выполняться как в режиме набора с клавиатуры, так и в программах. Данные импортируются или экспортируются из таблиц SQL в текущую БД. Внешние файлы могут быть в текущем или другом директории. В последнем случае, как обычно, следует перед именем файла указать имя диска и маршрут по директориям.

Для добавления данных из внешнего файла в конец существующей таблицы SQL в текущей БД используется команда

LOAD DATA FROM [путь]<имя файла>
INTO TABLE <имя таблицы>
[[TYPE]SDF/DIF/WKS/SYLK/FW2/RPD/DBASEII/
DELIMITED [WITH BLANK/WITH <разделитель>]];

Если не задан тип файла, то предполагается файл dBASE (кроме dBASE II). Для файлов dBASE импортируются данные только тех полей, которые определены как колонки в заданной таблице SQL (поля **Memo** не копируются).

При импорте из электронных таблиц или файлов с разделителями структура таблицы SQL, в которую загружаются данные, должна соответствовать формату файла, из которого осуществляется импорт. Определения колонок таблицы должны соответствовать длине и типу данных импортируемого файла. Если в импортируемом файле поля шире колонок таблицы, то происходит усечение не поместившихся данных. Строки электронной таблицы помещаются в строки таблицы SQL.

Копирование данных из таблиц SQL во внешние типы файлов осуществляется командой

UNLOAD DATA FROM [путь по директориям]<имя файла>
FROM TABLE <имя таблицы>
[[TYPE]SDF/DIF/WKS/SYLK/FW2/RPD/DBASEII/
DELIMITED [WITH BLANK/WITH <разделитель>]];

Здесь, так же как и для команды **LOAD**, не требуется указывать тип файла для файлов dBASE, кроме dBASE II. В экспортируемом файле создаются поля (колонки электронной таблицы) в соответствии со структурой таблицы SQL.

Если таблица защищена командой **PROTECT**, то необходимо иметь привилегию **SELECT**, чтобы выполнить команду **UNLOAD**. При этом файл автоматически дешифруется.

4.8 ЗАЩИТА ИНФОРМАЦИИ И РАБОТА В СРЕДЕ ЛВС

Информация в dBASE IV защищается как в режиме dBASE, так и в режиме SQL. Система защиты организуется и устанавливается администратором системы с использованием команды dBASE IV **PROTECT**. После назначения паролей и ввода данных по защите информации пользователи входят в систему по

индивидуальным паролям и выполняют те операции с данными, которые определены установленными привилегиями. В режиме dBASE IV привилегии по работе с файлами назначаются при выполнении команды **PROTECT**. В режиме SQL привилегии назначаются командами **GRANT** и **REVOKE**. После ввода данных по защите информации с помощью команды **PROTECT** доступ к файлам БД, определенным как таблицы SQL, осуществляется только командами SQL. Информация о пользователях, которые работают с таблицами SQL, введенная с помощью команды **PROTECT**, хранится в файле **Dbsystem.sql** в том же директории, что и аналогичный файл **Dbsystem.db** при операциях с файлами БД в режиме dBASE.

Перед выполнением команд **GRANT** и **REVOKE** в режиме SQL необходимо с помощью команды **PROTECT** ввести имена групп пользователей, их идентификаторы и пароли. Одному из пользователей может присваивается системный идентификатор - **SQLDBA**. Пользователь с этим идентификатором имеет привилегии администратора системы по выполнению любых операций SQL с любыми таблицами и выборками. Полный объем привилегий над созданными таблицами имеют их авторы. Они могут делегировать и ограничивать привилегии другим пользователям по работе со своими таблицами (пользователь с идентификатором **SQLDBA** имеет такие права по всем таблицам). Все привилегии, назначенные командами **GRANT** и **REVOKE**, записываются в таблицы каталога для активной БД.

Как и в режиме dBASE IV, в режиме SQL предусмотрено шифрование данных. Защищенные таблицы шифруются сразу после их создания. Данные в таблицах, созданных в режиме SQL, недоступны в режиме dBASE. Для обеспечения доступа к ним в режиме dBASE необходимо скопировать их из таблицы в файл БД командой **UNLOAD** или использовать предложение **SAVE TO TEMP** в команде **SELECT**.

Подобно этому нельзя выполнять команды **DBDEFINE** или **LOAD** для зашифрованных в режиме dBASE файлов БД. Перед выполнением этих команд надо предварительно установить состояние (выполнить команду) **SET ENCRYPTION OFF** и скопировать с помощью команды **COPY FILE** файлы, которые при копировании дешифруются. Для создания дешифрованной копии зашифрованного в режиме dBASE файла пользователю необходимо иметь привилегию по чтению этого файла.

Для назначения привилегий пользователям по работе с таблицами и выборками используется команда

GRANT ALL [PRIVILEGES]/<список привилегий>
ON [TABLE]<список таблиц>
TO PUBLIC/<список пользователей>
[WITH GRANT OPTION];

Эта команда позволяет назначить привилегии, которые соответствуют операциям над таблицами, выполняемыми командами **ALTER, CREATE, INDEX, DELETE, INSERT, SELECT** и **UPDATE.** Дополнительно назначаемые привилегии не отменяют имевшиеся, т.е. привилегии накапливаются при повторном выполнении команды **GRANT**.

Пользователю можно назначить привилегию делегировать все или часть своих привилегий другим пользователям (опция **WITH GRANT OPTION**). Если привилегии пользователя аннулируются, то соответственно аннулируются привилегии, делегированные этим пользователем другим пользователям.

Команда **GRANT** позволяет назначить все привилегии с помощью опции **ALL** или отдельные привилегии в виде соответствующих ключевых слов:

ALTER - возможность добавлять колонки в таблицу;

DELETE - возможность удалять строки из таблицы или выборки;

INDEX - возможность использовать команду **CREATE INDEX**;

INSERT - возможность добавлять строки в таблицу или выборку;

SELECT - возможность выводить строки из таблицы или выборки;

UPDATE [(список колонок)] - возможность обновлять все или заданные колонки в таблице.

В команде **GRANT** можно назначить заданные привилегии всем пользователям (опция **TO PUBLIC**) или пользователям, указанным в списке пользователей (список идентификаторов пользователей, введенных командой **PROTECT**).

Примеры:

```
GRANT ALL ON TABLE Kadry TO Ivanov;
GRANT INSERT, UPDATE, DELETE ON Plan TO PUBLIC
```

В первом примере предоставляются все привилегии для операций над таблицей Kadry пользователю с идентификатором Ivanov. Во втором примере предоставляются всем пользователям привилегии INSERT, UPDATE и DELETE для работы с таблицей Plan.

Аннулирование привилегий пользователям по работе с таблицами и выборками выполняется командой

REVOKE ALL [PRIVILEGES]/<список привилегий>
ON [TABLE] <список таблиц>
FROM PUBLIC/<список пользователей>;

Можно аннулировать все привилегии для указанных пользователей с помощью опции **ALL** или указать конкретные привилегии в виде ключевых слов, отделенных запятыми. Ключевые слова те же, что и в команде **GRANT**.

Пример:

```
REVOKE UPDATE, INSERT ON Plan From Petrov;
```

Команда **REVOKE** с опцией **PUBLIC** аннулирует только привилегии, предоставленные ранее командой **GRANT TO PUBLIC**, так как пользователи могут дополнительно получать привилегии, назначенные им индивидуально.

В режиме SQL, как и в dBASE IV, предусматривается три способа разрешения конфликтов, возникающих в ситуации одновременного обращения нескольких пользователей к одному файлу: установка файлов в режим обработки только одним пользователем, захват файлов в течение выполнения операции и захват отдельных записей.

В ряде случаев, например в ходе многочисленных обновлений данных всего файла БД, требуется установить исключительную прерогативу обработки этого файла одним пользователем, запретив обращение к нему других пользователей. В других случаях, например при выполнении небольшого объема обновлений, можно открыть доступ многих пользователей к файлу.

В режиме SQL, как и в dBASE IV, захват записей и файлов осуществляется автоматически. При выполнении команд SQL по вводу, обновлению и удалению данных в таблице или выборке SQL автоматически определяется необходимый вид захвата. Если невозможно завершить операцию захвата ввиду того, что информация захвачена другим пользователем, продолжается попытка захвата в течение заданного периода времени, определенного командой **SET REPROCESS**.

Для большей уверенности выполнения обновлений при обращении к разделенным данным, как и в режиме dBASE, можно использовать команды транзакции **BEGIN** и **END TRANSACTION**. Если операция транзакции не завершилась успешно, имеется возможность возвратить таблицы в исходное состояние, используя команду **ROLLBACK**.

Глава 5
ПРАКТИЧЕСКОЕ ПРИМЕНЕНИЕ ЯЗЫКА dBASE IV ПРИ РАЗРАБОТКЕ ПРИКЛАДНЫХ ПРОГРАММ

5.1. МОДУЛЬ ОРГАНИЗАЦИИ ВЫВОДА И ОБНОВЛЕНИЯ ДАННЫХ В ЗАДАЧЕ УЧЕТА ПЕРЕВОЗОК

В гл. 2 был рассмотрен пример создания простой информационной задачи учета перевозок автотранспортным предприятием с помощью средств УЦ. В настоящей главе покажем, как расширить эту задачу, дополнив ее модулем

обработки информации, который создан с использованием командного языка dBASE IV.

Модуль обработки информации по перевозкам с именем GRUZ позволяет пользователю вводить, добавлять и удалять данные в экранной форме, представляющей собой карточку на водителя. В этой карточке, показанной ниже, высвечиваются данные из трех файлов БД с именами BD1, BD2, BD3, структура которых приведена в гл. 2.

```
 Записи     Перейти     Выход                            8:31:41pm
ВОДИТЕЛЬ: Строков   Олег    Валерьевич    Дата рождения 05/08/65

 Категория: B,C      Стаж работы:  4

АВТОМОБИЛЬ:ГАЗ-53 Номерной знак 34-56 ММД  Находится в автопарке
                  Техническое состояние: исправен

Станки ЧПУ      | 3.0|Одинцово    |24/03/89|Горький     |25/03/8
Мебель          | 1.0|Москва      |30/06/89|Калуга      |01/07/8
Двигатели       | 3.0|Москва      |12/10/89|Рязань      |14/10/8

F9:Таб|F10:Меню|PgDn:След.|PgUp:Пред.|Ctrl-PgDn:Посл.|Ctrl-PgUp:Пер
```

Экранная форма карточки состоит из верхней и нижней частей. В верхней части выводятся поля из файлов BD1 и BD2, содержащие данные о водителе и его автомобиле, а в нижней (в виде стандартной формы Browse) - поля из файла BD3, содержащие данные о перевозках, включая наименование груза, его вес, исходный и конечный пункты перевозки, дату выезда и прибытия, фактическое время нахождения в пути и расход горючего. Последние два поля не видны на рисунке, так как они расположены в скрытой правой части экранной формы Browse.

При эксплуатации информационной задачи учета перевозок требуется вводить новые данные в карточки водителей (данные о выполненных перевозках, состоянии автомобиля и т.д.), а также заводить новые карточки на вновь принятых водителей и удалять карточки уволенных водителей. После вызова модуля GRUZ высвечивается карточка водителя.

Для обеспечения перечисленных операций служит основное (горизонтальное) меню с позициями "Записи", "Перейти" и "Выход", находящееся в верхней части экрана карточки. Это меню активизируется нажатием функциональной клавиши F10. Ниже показаны позиции основного меню и прикрепляемые к ним вертикальные меню.

Чтобы найти нужную карточку, можно перелистать их вручную клавишами PgDn (вперед) и PgUp (назад) или же использовать позицию "Перейти". При выборе этой позиции высвечивается вертикальное меню с позициями "Первая карточка", "Последняя карточка", "Карточка с номером", "Пропустить", "Поиск по фамилии".

Записи	Перейти	Выход
Добавить новые карточки	Первая карточка	Выход из задачи
	Последняя карточка	
Пометить карточку для удаления	Карточка с номером	
Удалить помеченные карточки	Пропустить	
	Поиск по фамилии	

Первые две позиции позволяют перейти соответственно к первой и последней карточкам водителей, фамилии которых упорядочены по алфавиту. Следующая позиция позволяет выбрать водителя по номеру его карточки. После выбора этой позиции высвечивается поле для ввода номера карточки. Для высвечивания карточки с заданным номером нужно по окончании ввода номера нажать клавишу Enter. Для пропуска вперед или назад нескольких карточек нужно после выбора позиции "Пропустить" ввести число пропускаемых карточек (со знаком "плюс" - вперед, а со знаком "минус" - назад). Высвечивание карточки по фамилии осуществляется выбором позиции "Поиск по фамилии", после чего высвечивается окно, в которое нужно ввести требуемую фамилию. Можно вводить фамилию не полностью, а лишь то число букв, начиная с первой, которое однозначно идентифицирует фамилию. Если в результате поиска требуемая карточка не найдена, то выдается сообщение "Такой карточки НЕТ!!!".

Если же карточка найдена, она высвечивается на экране и с ней можно выполнять операции. Переход по полям карточки осуществляется клавишей Tab, переход из верхней части карточки в нижнюю - клавишей F9, а обратно - клавишей Esc. Обновление введенных данных (например, состояния автомобиля) осуществляется посредством установки курсора на нужное поле и изменения содержимого поля с клавиатуры. Фиксация исправлений выполняется клавишей Enter.

Удаление карточки реализуется в два этапа. Сначала нужно установить курсор на фамилию водителя и пометить ее, выбрав позицию "Пометить карточку для удаления". Таким способом можно пометить несколько карточек. Признаком пометки карточки является то, что вместо текста позиции "Пометить карточку для удаления" будет высвечиваться текст "Убрать пометку". Выбор последней позиции приводит к уничтожению пометки.

Для того чтобы завести новую карточку, используется позиция "Добавить новую карточку" этого же меню. После ее выбора высвечивается пустая экранная форма ввода данных.

В модуле предусмотрена возможность получить в любой момент работы справочную информацию: полные сведения о клавишах, которые используются при работе с экранной формой (краткая справка о клавишах всегда высвечивается в нижней части экрана). Эта информация вызывается клавишей F1. К справочной информации также относится сообщение - комментарий, высвечиваемый в нижней строке экрана при выборе позиций меню. При высвечивании полной справочной информации экран имеет вид, показанный ниже.

В данном разделе приводится текст модуля GRUZ с пояснениями о работе процедур и отдельных команд. Модуль GRUZ состоит из главной процедуры и вызываемых из нее процедур.

В начале главной процедуры с помощью SET-команд выполняются установки параметров системы dBASE IV. Затем определяются окна с именами Table, Pause и Scr_hlp для высвечивания соответственно таблицы с данными о перевозках в нижней части карточки водителя, сообщений об ошибках при выполнении программы и справочной информации. Команда ON ERROR при возникновении ошибки вызывает процедуру Pause. Этой процедуре передается параметр, содержащий имя процедуры и номер строки, в которой произошла ошибка.

```
Записи      Перейти      Выход                          8:31:41pm

ВОДИТЕЛЬ: Строков   Олег    Валерьевич    Дата рождения 05/08/65

      Навигационные клавиши при работе с экранной формой:
     --------------------------------------------------
     F1:  Вызов справочной информации
     F9:  Переход из верхней в нижнюю часть экранной формы
     Esc: Переход из нижней в верхнюю часть экранной формы
     F10: Активизация меню  экранной формы
     PgDn: Переход к следующей карточке
     PgUp: Переход к предыдущей карточке
     Ctrl-PgDn: Переход к последней карточке
     Ctrl-PgUp: Переход к первой карточке

    Нажмите любую клавишу ...

Станки ЧПУ      | 3.0|Одинцово     |24/03/89|Горький      |25/03/8
Мебель          | 1.0|Москва       |30/06/89|Калуга       |01/07/8
Двигатели       | 3.0|Москва       |12/10/89|Рязань       |14/10/8

F9:Таб|F10:Меню|PgDn:След.|PgUp:Пред.|Ctrl-PgDn:Посл.|Ctrl-PgUp:Пер
```

Команда RESTORE MACROS FROM вызывает макроопределения с именами Menu, Exit_brow и Lineitem из файла макроопределений TAB.key. Эти макроопределения созданы предварительно средствами УЦ и предназначены для имитации нажатий клавиш. Макроопределение Menu предназначено для вызова клавишей F10 основного (горизонтального) меню. Это макроопределение имитирует нажатия последовательности клавиш Ctrl-End (сохранение текущей записи) и Ctrl-F10 (передача в опцию CASE десятичного кода клавиши, служащего для идентификации вызова меню). Заметим, что вместо последней комбинации клавиш можно, в принципе, использовать любую комбинацию или отдельную клавишу, поставив ее десятичный код в опцию CASE.

Макроопределение Lineitem предназначено для обеспечения перехода из верхней части карточки водителя в нижнюю при нажатии клавиши F9. Оно включает имитацию последовательности клавиш Ctrl-End (сохранение текущей записи) и Ctrl-F9. Функция последней комбинации аналогична комбинации Ctrl-F10 из предыдущего макроопределения. Макроопределение Exit_brow служит для вывода таблицы Browse в нижней части экранной формы и перехода в верхнюю часть и выполняется в процедуре Showbrow при высвечивании карточки водителя. Это макроопределение имитирует нажатие клавиши Esc.

```
CLEAR
SET COLOR OF MESSAGE to W+/BR
SET COLOR OF BOX to N/BR
SET COLOR OF INFORMATION to N/BR
SET CLOCK ON
SET DELIMITERS OFF
SET ECHO OFF
SET TALK OFF
SET ECHO OFF
SET SCOREBOARD OFF
SET STATUS ON

DEFINE WINDOW Table FROM 12,2 TO 21,77 NONE COLOR gr+/n
```

```
DEFINE WINDOW Pause FROM 14,01 TO 19,78 NONE COLOR W+/R+
DEFINE WINDOW Scr_hlp FROM 5,00 TO 20,79 NONE COLOR GR+/B

ON ERROR DO Pause WITH "Программа "+Program()+": "+"Строка :"+;
             LTRIM(STR(LINE()))
RESTORE MACROS FROM TAB

```

Далее команды USE открывают файлы БД BD1, BD2 и BD3 в рабочих областях 1, 2 и 3. Команды ON KEY назначают клавишам F1, F9, F10 выполнение процедуры Scr_hlp и макроопределений Lineitem и Menu. Процедура Defnmenu вызывается для определения основного горизонтального меню модуля GRUZ и прикрепленных к его позициям вертикальных подменю.

Символьной переменной gc_messg присваивается текст краткой справочной информации об используемых клавишах, который будет высвечиваться в строке сообщений экрана. Другим переменным присваиваются начальные значения, и после очистки 23-й строки экрана в ней высвечивается текст сообщения о функциях клавиш, записанный в переменной gc_messg.

```
SELECT 3
USE BD3 ORDER n_z
SET CARRY TO n_z
SELECT 2
USE BD2.DBF ORDER n_z
SELECT 1
USE BD1.DBF ORDER fam

ON KEY LABEL F1 DO Scr_hlp
ON KEY LABEL F9 PLAY MACRO Lineitem
ON KEY LABEL F10 PLAY MACRO Menu
DO Defnmenu
gc_messg ="F9:Табл.|F10:Меню|PgDn:След.|PgUp:Пред.";
      +"|Ctrl-PgDn:Послед.|Ctrl-PgUp:Первая"
gn_gorec = 0
gc_mdx =MDX(1)
gc_search = SPACE(200)
gc_seek = SPACE(LEN(fam))
gl_newrec = .f.
gl_chgrec = .t.
gl_extloop = .f.

@ 23,0
@ 23,2 SAY gc_messg

```

Основой главной процедуры служит вложенная структура двух команд цикла DO WHILE...ENDDO. Выход из внешнего цикла осуществляется при присвоении переменной gl_extloop значения .t. в процедуре Get_exit, которая вызывается при выборе позиции "Выход из задачи". Во внутреннем цикле выполняется разветвление управления в зависимости от значения переменной gn_inkey или gn_readkey. В переменную gn_inkey заносится значение десятичного кода последней нажатой клавиши или комбинации клавиш, возвращаемые функцией INKEY(). При значении переменной gn_inkey=-19 (десятичный код комбинации клавиш Ctrl-F10, который служит признаком для высвечивания меню) выполняется активизация меню задачи. При значении этой переменной -18 осуществляется переход из верхней в нижнюю часть карточки водителя. В переменную gn_readkey заносятся значения десятичного кода клавиши или комбинации клавиш, которые возвращаются функцией READKEY() и нажаты пользователем в таблице Browse.

Например, при нажатии клавиш PgUp осуществляется переход к предыдущей карточке, а при нажатии клавиш PgDn - переход к следующей карточке. При этом если карточка последняя, то высвечивается предложение создать новую карточку и вызывается процедура Recappend, которая реализует операции по высвечиванию пустой карточки. Аналогично при нажатии клавиш Ctrl-PgUp или Ctrl-PgDn осуществляется переход к первой или последней карточке, а при нажатии клавиш Esc или Ctrl-End - переход из таблицы Browse в верхнюю часть карточки.

В завершающей части главной процедуры команда ON KEY восстанавливает назначение клавишам F1, F9, F10 функций, заданных по умолчанию в dBASE IV, а команда ON ERROR - вывод системных сообщений об ошибках dBASE IV. Команды RELEASE удаляют из оперативной памяти все меню, окна, экран и переменные.

```
DO WHILE .NOT. gl_extloop
  gn_recno = RECNO()
  gl_lineitm = .T.
  DO WHILE gl_lineitm
     IF gl_chgrec
       DO Show_get
     ENDIF
     DO Showbrow
     DO Edit_get
     READ
     replace bd2->n_z with bd1->n_z

     gn_inkey = INKEY()
     gn_readkey = READKEY()
     gc_readvar = VARREAD()
     gl_lineitm = .F.
     ACTIVATE SCREEN
     DO CASE
       CASE gn_inkey = -18
        DO Lineitem
        gl_lineitm = .T.
       CASE gn_inkey = -19
        ACTIVATE MENU Editmenu
       CASE gn_readkey = 6  .OR. gn_readkey = 262 .OR.;
           gn_readkey = 260.OR. gn_readkey = 4
        IF .NOT. BOF()
          SKIP -1
        ENDIF
       CASE gn_readkey = 7 .OR. gn_readkey = 263 .OR.;
           gn_readkey = 5 .OR. gn_readkey = 261
        SKIP
        IF EOF() .AND. .NOT. gl_newrec
          CLEAR GETS
         @ 23,0
         @ 23,22 SAY "=> Добавить новые карточки (Y/N)? <=" ;
              GET gl_newrec PICT "Y"
         READ
         CLEAR GETS
         @ 23,0
         @ 23,2 SAY gc_messg
       ENDIF
```

```
      IF gl_newrec
        DO Recappnd
      ELSE
        IF EOF()
          SKIP -1
        ENDIF
      ENDIF
    CASE gn_readkey = 34 .OR. gn_readkey = 290
      GO TOP
    CASE gn_readkey = 35 .OR. gn_readkey = 291
      GO BOTTOM
    CASE gn_readkey = 12 .or. gn_readkey = 270
      gl_extloop = .T.
      EXIT
  ENDCASE
  gl_chgrec = IIF(gn_recno = RECNO(), .F., .T.)
 ENDDO
DO Chkdele
ENDDO
ON KEY
ON ERROR
SELECT 1
CLOSE ALL
RELEASE MENU Editmenu
RELEASE POPUPS Records, Go_to, Exit
RELEASE WINDOWS Table,Pause,Seek,Search,Bsearch,Scr_hlp
RELEASE lc_talk,lc_fields,lc_status,lc_escape
RETURN
```

Из главной процедуры вызываются следующие процедуры: Show_get, Edit_get, Showbrow, Lineitem, Keyset, Keykill, Chkdele, Recappnd, Get_recs, Get_goto, Get_exit, Pause, Scr_hlp, Defnmenu.

Процедура **Defnmenu** определяет основное (горизонтальное) меню данного модуля с именем Editmenu и прикрепленные к его позициям вертикальные меню с именами Records, Go_to, Exit, а также окно с именем Seek, в которое вводится фамилия при поиске карточки.

Эти меню позволяют выполнять различные операции над записями БД, а по окончанию работы выйти в УЦ. В начале процедуры определяется горизонтальное меню с именем Editmenu, состоящее из трех позиций "Записи", "Перейти" и "Выход", расположенных в нулевой строке экрана. К позициям горизонтального меню прикрепляются вертикальные меню. Командами ON PAD назначается активизация вертикальных меню с именами Records, Go_to или Exit при выборе позиций горизонтального меню с теми же именами.

Далее определяется вертикальное меню с именем Records, имеющее три активные позиции. Это меню выполняет действия по добавлению новой карточки в файл БД, пометке для удаления или отмены пометки текущей карточки, а также удаление помеченных карточек.

В следующих строках определяется вертикальное меню с именем Go_to. Это вертикальное меню реализует функции поиска первой, последней и карточки с заданным номером, пропуск вперед и назад требуемого числа карточек и поиск карточки по фамилии.

Вертикальное меню Exit состоит из одной позиции, при выборе которой осуществляется выход из задачи.

Командами ON SELECTION POPUP определяются процедуры, которые будут выполняться при активизации вертикальных меню. При активизации меню с именами Records, Go_to и Exit выполняются соответственно процедуры с именами Get_recs, Get_goto и Get_exit.

В конце процедуры **Defnmenu** команда DEFINE WINDOW определяет окно с именем Seek, а команда SHOW MENU высвечивает в верхней строке экрана позиции горизонтального меню Editmenu, но не активизирует его.

```
PROCEDURE Defnmenu
SET BORDER TO SINGLE
DEFINE MENU Editmenu
DEFINE PAD Records OF Editmenu PROMPT "Записи " AT 0,2
DEFINE PAD Go_to   OF Editmenu PROMPT "Перейти" AT 0,14
DEFINE PAD Exit    OF Editmenu PROMPT "Выход  " AT 0,26
ON PAD Records OF Editmenu ACTIVATE POPUP Records
ON PAD Go_to   OF Editmenu ACTIVATE POPUP Go_to
ON PAD Exit    OF Editmenu ACTIVATE POPUP Exit
DEFINE POPUP Records FROM 2,0
DEFINE BAR 1 OF Records PROMPT " Добавить новые карточки   "
DEFINE BAR 2 OF Records PROMPT "──────────────────────────" SKIP
DO Chkdele
DEFINE BAR 4 OF Records PROMPT " Удалить помеченные карточки";
MESSAGE "Физическое удаление помеченных карточек в БД"
DEFINE POPUP Go_to FROM 2,12
DEFINE BAR 1 OF Go_to PROMPT "   Первая карточка";
MESSAGE "Перейти к первой карточке БД"
DEFINE BAR 2 OF Go_to PROMPT "   Последняя карточка";
MESSAGE "Перейти к последней карточке БД"
DEFINE BAR 3 OF Go_to PROMPT "   Карточка номер";
MESSAGE "Перейти к карточке с указанным номером"
DEFINE BAR 4 OF Go_to PROMPT "   Пропустить";
MESSAGE 'Число пропускаемых карточек (знак "-" - переход назад)'
DEFINE BAR 5 OF Go_to PROMPT "──────────────────────────" SKIP
DEFINE BAR 6 OF Go_to PROMPT "   Быстрый поиск по фамилии  " ;
MESSAGE "Применяется для поиска по индексу при заданном значении"
DEFINE POPUP Exit FROM 2,22
DEFINE BAR 1 OF Exit PROMPT "  Выход из задачи учета перевозок  ";
MESSAGE "Сохранить изменения в текущей карточке и выйти из задачи"
ON SELECTION POPUP Records DO Get_recs
ON SELECTION POPUP Go_to DO Get_goto
ON SELECTION POPUP Exit DO Get_exit
DEFINE WINDOW Seek FROM 8,15 TO 11,44
DEFINE WINDOW Search FROM 9,15 TO 11,60
DEFINE WINDOW Bsearch FROM 10,15 TO 12,60
SHOW MENU Editmenu
RETURN
```

Процедура **Chkdele** вызывается при активизации вертикального меню Records и высвечивает различный текст во второй его позиции в зависимости от того, помечена ли текущая карточка. Если она помечена для удаления, то в этой позиции меню высвечивается текст "Убрать пометку". Если же карточка не

помечена для удаления, то высвечивается текст "Пометить карточку для удаления".

```
PROCEDURE Chkdele
IF DELETED()
   DEFINE BAR 3 OF Records PROMPT "  Убрать пометку ";
   MESSAGE "Восстановление карточки, помеченной для удаления"
ELSE
   DEFINE BAR 3 OF Records PROMPT " Пометить карточку для удаления";
   MESSAGE "Пометка карточки, предназначенной для удаления"
ENDIF
RETURN
```

Процедура **Showbrow** обеспечивает вывод в таблице Browse тех записей о перевозках, которые выполнены водителем, карточка которого высвечена на экране. Вывод таких записей реализуется командой SET FILTER, которая в файле БД BD3 отбирает записи, относящиеся к этому водителю. Процедура только высвечивает записи в таблице Browse, но не дает возможности их редактировать.

```

PROCEDURE Showbrow
SELECT 3
SET FILTER TO
SEEK bd1->n_z
IF EOF()
   APPEND BLANK
   REPLACE n_z WITH bd1->n_z
ENDIF
SET FILTER TO n_z = bd1->n_z
GO TOP
PLAY MACRO Exit_brow
BROWSE FIELDS n_g,ves,o_p,d_v,k_d,d_p,vrm,rgr,n_z ;
WINDOW Table COMPRESS NOMENU NOCLEAR
SELECT 1
ACTIVATE SCREEN
RETURN

```

Процедура **Lineitem** в отличие от процедуры Showbrow предоставляет возможность пользователю в таблице Browse редактировать записи по перевозкам, выполненным водителем, и вызывается после нажатия функциональной клавиши F9 (переход в нижнюю часть карточки водителя). В начале этой процедуры посредством вызова процедуры Keykill осуществляется подавление функций, ранее назначенных клавишам F1, F9 и F10. Заметим, что функции этих клавиш, заданные по умолчанию в dBASE IV, не восстанавливаются, т.е. они просто закрываются для пользователя.

```
PROCEDURE Lineitem
ln_key = INKEY()
DO Keykill
SELECT 3
IF EOF()
   APPEND BLANK
   REPLACE n_z WITH bd1->n_z
ENDIF
SET FILTER TO n_z = bd1->n_z
GO TOP
BROWSE FIELDS n_g,ves,o_p,d_v,k_d,d_p,vrm,rgr,n_z ;
```

```
WINDOW Table COMPRESS NOMENU NOCLEAR
ACTIVATE SCREEN
DO Keyset
SELECT 1
RETURN

```

Процедура **Keyset** осуществляет назначение функций клавишам F1, F9, F10, которые предусмотрены в данном модуле в отличие от принятых по умолчанию в dBASE IV.

```
PROCEDURE Keyset
ON KEY LABEL F1 DO Scr_hlp
ON KEY LABEL F9 PLAY MACRO Lineitem
ON KEY LABEL F10 PLAY MACRO Menu
RETURN

```

Процедура **Keykill** отменяет функции, которые были назначены функциональным клавишам F1, F9, F10, посредством выполнения фиктивной команды - занесения в переменную ln_key значения нажатой клавиши.

```
PROCEDURE Keykill
ON KEY LABEL F1 ln_key=INKEY()
ON KEY LABEL F9  ln_key=INKEY()
ON KEY LABEL F10  ln_key=INKEY()
RETURN

```

Процедура **Recappnd** осуществляет переход к последней карточке БД, добавляет в конец файлов БД BD1 и BD2 пустую запись и высвечивает пустую карточку водителя.

```
PROCEDURE Recappnd
APPEND BLANK
gl_newrec=.f.
RETURN

```

Процедура **Get_recs** обеспечивает реализацию функций вертикального меню Records, имеющего три активные позиции: "Добавить новую карточку", "Пометить карточку для удаления" ("Убрать пометку") и "Удалить помеченные карточки".

```
PROCEDURE Get_recs
DO CASE
  CASE BAR() = 1 && добавление
  DO Recappnd
  CASE BAR() = 3 && пометка/восстановление
    IF DELETED()
      RECALL
      SELECT 3
      RECALL ALL
      SELECT 2
      RECALL
    ELSE
      DELETE
      SELECT 3
```

```
      DELETE ALL
      SELECT 2
      DELETE
    ENDIF
    SELECT 1
  CASE BAR() = 4  && удаление
    PACK
    SELECT 3
    PACK
    SELECT 2
    PACK
    SET ORDER TO N_Z
ENDCASE
DO Chkdele
DEACTIVATE MENU
RETURN

```

Процедура **Get_goto** реализует функции вертикального меню Goto по переходу к первой и последней записи файла БД, записи с заданным номером, пропуск вперед и назад заданное число записей, а также поиск по фамилии. Разветвление управления реализуется с помощью команды DO CASE... ENDCASE. Первые две функции выполняются командами GO TOP и GO BOTTOM, расположенными после соответствующих опций CASE. При поиске по номеру карточки в команду @...GET включена опция RANGE для проверки того, не выходит ли номер вводимой карточки за пределы первой и последней записи. Если выходит, то высвечивается системное сообщение dBASE IV, в котором указывается заданный диапазон.

Для реализации функции пропуска заданного числа карточек при выборе позиции "Пропустить" введенное с клавиатуры число заносится в переменную skiprec, которая затем используется в команде SKIP.

При реализации функции поиска карточки по фамилии введенная с клавиатуры фамилия заносится в переменную gc_seek, которая используется вместе с командой SEEK. Если в файле БД нет водителя с такой фамилией, вызывается процедура Pause и в окне высвечивается сообщение "Такой фамилии в списке нет !!!".

```
PROCEDURE Get_goto
mpict = REPLICATE("9", LEN( LTRIM( STR( RECCOUNT()))))
gc_search = Gc_search + SPACE(200 - LEN(Gc_search))
gc_seek = Gc_seek + SPACE(LEN(FAM) - LEN(Gc_seek))
DO CASE
  CASE BAR() = 1
    GO TOP
  CASE BAR() = 2
    GO BOTTOM
  CASE BAR() = 3
    @ 5,39 GET gn_gorec RANGE 1, RECCOUNT() PICTURE Mpict
    READ
    gn_gorec = IIF( Gn_gorec=0, RECNO(), Gn_gorec)
    GO gn_gorec
  CASE BAR() = 4
    skiprec = IIF( RECCOUNT() > 9, 10, 5)
    @ 5,38 GET skiprec PICTURE Mpict
    READ
    SKIP skiprec
    IF EOF()
      SKIP -1
```

```
  ENDIF
 CASE BAR() = 6
  ACTIVATE WINDOW Seek
  IF "" = ORDER()
    tempmdx = TAG(2)
    SET ORDER TO TAG &tempmdx.
  ENDIF
  @ 0,1 SAY "Введите фамилию водителя"
  @ 1,5 GET gc_seek PICT "@S20" MESSAGE "Отменить: Esc"
  READ
  IF READKEY() <> 12
    gc_seek = LTRIM( TRIM(gc_seek))
    mrec = RECNO()
    SEEK gc_seek
    IF .NOT. FOUND()
      DO Pause WITH "Такой фамилии в списке нет !!!"
      GO mrec
    ENDIF
  ENDIF
  DEACTIVATE WINDOW Seek
  ACTIVATE SCREEN
ENDCASE
DEACTIVATE MENU
RETURN

```

Процедура **Get_exit** вызывается при выборе позиции вертикального меню Exit. В ней выполняется команда CLEAR GETS по очистке полей переменных команд GET, затем переменной gl_extloop присваивается значение .T., для обеспечения выхода из цикла DO WHILE...ENDDO в главной процедуре и деактивизируется основное горизонтальное меню Editmenu.

```
PROCEDURE Get_exit
CLEAR GETS
gl_extloop = .T.
DEACTIVATE MENU
RETURN

```

Процедура **Pause** вызывается при возникновении ошибки в программе и выводит в окне Pause информацию о характере ошибки, имя процедуры и номер строки, в которой она произошла.

```
PROCEDURE Pause
PARAMETER lc_msg
IF TYPE("lc_message")="U"
  Gn_error=ERROR()
ENDIF
lc_msg = lc_msg
lc_option='0'
ACTIVATE WINDOW Pause
IF gn_error > 0
  IF TYPE("lc_message")="U"
    @ 0,1 SAY [Произошла ошибка !! - Сообщение: ]+MESSAGE()
  ELSE
    @ 0,1 SAY [Ошибка # ]+lc_message
  ENDIF
ENDIF
@ 2,10 SAY lc_msg
```

```
WAIT " ... нажмите любую клавишу "
DEACTIVATE WINDOW Pause
RETURN
```

Процедура **Scr_hlp** вызывается при нажатии клавиши F1 и высвечивает в окне Hlp справочную информацию о клавишах, используемых в задаче. Сначала высвечивается текст, объясняющий использование клавиш при работе в верхней части карточки водителя, затем после нажатия любой клавиши в окне появляется информация о клавишах, используемых при работе с таблицей Browse в нижней части карточки водителя.

В этой процедуре используется команда TEXT...ENDTEXT, позволящая вывести на экран текст, введенный в программу.

```
PROCEDURE Scr_hlp
gc_readvar = VARREAD()
ACTIVATE WINDOW Scr_hlp
CLEAR
TEXT
        Навигационные клавиши при работе с экранной формой:
        ---------------------------------------------------

         F1:  Вызов текста экрана помощи
         F9:  Переход из экранной формы в таблицу
         F10: Активизация меню для экранной формы

         PgDn: Переход к следующей записи экранной формы
         PgUp: Переход к предыдущей записи экранной формы
         Ctrl-PgDn: Переход к последней записи БД, поля
                    которых размещены в экранной форме
         Ctrl-PgUp: Переход к первой записи БД

         Esc: Выход из задачи
ENDTEXT
@ 14,5 say "Нажмите любую клавишу ..."
x = INKEY(0)
CLEAR
TEXT
         Навигационные клавиши при работе с таблицей:
         ----------------------------------------------

            F1: Вызов текста экрана помощи

            PgDn: Переход к следующему экрану в таблице
            PgUp: Переход к предыдущему экрану в таблице
            Ctrl-PgDn: Переход к последней строке таблицы
            Ctrl-PgUp: Переход на первую строку таблицы

            Esc: Возвращение в экранную форму
ENDTEXT
@ 14,5 say "Нажмите любую клавишу..."
x = INKEY(0)
DEACTIVATE WINDOW Scr_hlp
ACTIVATE SCREEN
RETURN
```

Процедура **Show_get** создает верхнюю часть экрана карточки водителя и обеспечивает форматированный вывод полей из файлов баз данных BD1, BD2 с пояснительными записями.

```
PROCEDURE Show_get
@ 2,0  SAY " "
@ 2,2  SAY "┌──────────────────────────────────────────────";
       +"────────────────────────────────┐" COLOR n/r
@ 3,2  SAY "│" COLOR n/r
@ 3,3  SAY " " COLOR gr+/r
@ 3,4  SAY "ВОДИТЕЛЬ:" COLOR w+/r
@ 3,13 SAY " " COLOR gr+/r
@ 3,14 GET bd1->fam PICTURE "XXXXXXXXXXXXXXX" COLOR n/bg,n/bg
@ 3,29 SAY " " COLOR gr+/r
@ 3,30 GET bd1->nam PICTURE "XXXXXXXXXXXXXXX" COLOR n/bg,n/bg
@ 3,45 SAY " " COLOR gr+/r
@ 3,46 GET bd1->otc PICTURE "XXXXXXXXXXXXXXX" COLOR n/bg,n/bg
@ 3,61 SAY "  г.р. " COLOR gr+/r
@ 3,68 GET bd1->d_r COLOR n/bg,n/bg
@ 3,76 SAY " " COLOR gr+/r
@ 3,77 SAY "│" COLOR n/r
@ 4,2  SAY "│" COLOR n/r
@ 4,3  SAY "                                        ";
       +"                    " COLOR gr+/r
@ 4,77 SAY "│" COLOR n/r
@ 5,2  SAY "│" COLOR n/r
@ 5,3  SAY " Категория: " COLOR gr+/r
@ 5,15 GET bd1->kat PICTURE "XXXXXXXXX" COLOR n/bg,n/bg
@ 5,24 SAY "   Стаж работы: " COLOR gr+/r
@ 5,40 GET bd1->s_r PICTURE "99" COLOR n/bg,n/bg
@ 5,42 SAY "                                   " COLOR gr+/r
@ 5,77 SAY "│" COLOR n/r
@ 6,2  SAY "│" COLOR n/r
@ 6,3  SAY "                                        ";
       +"                    " COLOR gr+/r
@ 6,77 SAY "│" COLOR n/r
@ 7,2  SAY "│" COLOR n/r
@ 7,3  SAY "                                        ";
       +"                    " COLOR gr+/r
@ 7,77 SAY "│" COLOR n/r
@ 8,2  SAY "│" COLOR n/r
@ 8,3  SAY " " COLOR gr+/r
@ 8,4  SAY "АВТОМОБИЛЬ:" COLOR bg+/r
@ 8,15 SAY " " COLOR gr+/r
@ 8,26 SAY "   номерной знак " COLOR gr+/r
@ 8,43 GET bd1->n_z PICTURE "XXXXXXXXX" COLOR n/bg,n/bg
@ 8,52 SAY "  находится " COLOR gr+/r
DO Base2
@ 8,16 GET mar PICTURE "XXXXXXXXXX" COLOR n/bg,n/bg
@ 8,64 GET g_n PICTURE "XXXXXXXXXXXX" COLOR n/bg,n/bg
@ 9,52 GET sos PICTURE "XXXXXXXXXXXX" COLOR n/bg,n/bg
@ 8,76 SAY " " COLOR gr+/r
@ 8,77 SAY "│" COLOR n/r
@ 9,2  SAY "│" COLOR n/r
@ 9,9  SAY "                    техническое состояние: " COLOR gr+/r
@ 9,64 SAY "             " COLOR gr+/r
@ 9,77 SAY "│" COLOR n/r
@ 10,2 SAY "└──────────────────────────────────────────────";
       +"────────────────────────────────┘" COLOR n/r
```

```
CLEAR GETS
SELECT 1
RETURN

```

Процедура **Edit_get** высвечивает поля файлов баз данных BD1 и BD2. В главной процедуре после команды вызова процедуры **Edit_get** следует команда READ, активизирующая команды @...GET, находящиеся в процедуре, в результате чего становится возможным вводить и изменять высвечиваемые поля.

```
PROCEDURE Edit_get
@ 3,14 GET bd1->fam PICTURE "XXXXXXXXXXXXXXX" COLOR n/bg,n/bg
@ 3,30 GET bd1->nam PICTURE "XXXXXXXXXXXXXXX" COLOR n/bg,n/bg
@ 3,46 GET bd1->otc PICTURE "XXXXXXXXXXXXXXX" COLOR n/bg,n/bg
@ 3,68 GET bd1->d_r COLOR n/bg,n/bg
@ 5,15 GET bd1->kat PICTURE "XXXXXXXXX" COLOR n/bg,n/bg
@ 5,40 GET bd1->s_r PICTURE "99" COLOR n/bg,n/bg
@ 8,43 GET bd1->n_z PICTURE "XXXXXXXXX" COLOR n/bg,n/bg
DO Base2
@ 8,16 GET mar PICTURE "XXXXXXXXXX" COLOR n/bg,n/bg
@ 8,64 GET g_n PICTURE "XXXXXXXXXXXX" COLOR n/bg,n/bg
@ 9,52 GET sos PICTURE "XXXXXXXXXXXX" COLOR n/bg,n/bg
SELECT 1
RETURN

```

Процедура **Base2** служит для установки поля n_z в качестве управляющего индекса, поиска записи в файле BD2 с номером знака, выбранным из этого поля в текущей записи файла BD1, или занесения этого номера, если такой записи нет.

```
PROCEDURE Base2
SELECT 2
SET ORDER TO n_z
SEEK bd1->n_z
IF EOF()
   APPEND BLANK
   REPLACE n_z WITH bd1->n_z
ENDIF
RETURN
```

5.2. БИБЛИОТЕЧКА ПРОЦЕДУР, РЕАЛИЗУЮЩИХ СТАНДАРТНЫЕ ФУНКЦИИ

В данном разделе описывается библиотечка процедур, которые можно использовать (возможно, с некоторыми добавлениями и изменениями) в качестве элементов для построения прикладных программ. В библиотечку входят следующие процедуры: **Output.prg**, **Input.prg**, **Key.prg**, **Help.prg**, **Mnuprint.prg**, **Error.prg**, **Append.prg**, **Menu.prg**, **Music.prg**. Ниже приводится их описание вместе с текстом команд.

Процедура **Output** позволяет отобрать по заданному условию данные из файла БД и вывести их порциями на экран. После вывода очередной порции пользователю предоставляется возможность продолжить просмотр отобранных данных или прекратить вывод и выйти из программы.

Отбор данных осуществляется из файла базы данных BD1, который рассматривался в задаче учета перевозок в гл. 2, по полю даты рождения,

а на экран выводятся поля fam (фамилия), nam (имя), otc (отчество), d _ r (дата рождения).

В начале процедуры команда SET ECHO OFF подавляет вывод на экран выполняемых команд, а команда SET TALK OFF отменяет вывод результатов выполнения команд. Команда SET DATE GERMAN устанавливает европейский формат даты - ДД.ММ.ГГ, где ДД - число месяца, ММ - номер месяца, ГГ - год.

Далее переменным pdat1 и pdat2 присваиваются начальные значения pdat1 = {01.01.35} и pdat2 = DATE(), первой из них - значение некоторой даты, а второй - значение текущей даты, в результате чего тип этих переменных становится датой.

Затем создается экранная форма для ввода условия отбора данных. Команда @ 7,22 TO 15,52 рисует на экране рамку, @..SAY высвечивает пояснительный текст: "ВВЕДИТЕ", "-начальную дату:", "-конечную дату:", а две команды @...GET выводят начальные значения pdat1, pdat2 интервала отбора записей по дате рождения, которые заменяются требуемыми.

Команда READ активизирует команды @...GET, т.е. обеспечивает запись в переменные pdat1, pdat2 значений, которые вводятся с клавиатуры. Команда USE открывает файл БД с именем BD1, из которого в соответствии с заданным с помощью команды SET FILTER TO d _ r >= pdat1 .AND. d _ r <= pdat2 фильтром отбираются требуемые записи.

После фильтрации командой GO TOP осуществляется переход на первую отобранную запись. Команда SET FIELDS TO fam,nam,otc,d _ r определяет перечень выводимых на экран полей, а CLEAR очищает экран.

Вывод отобранных записей на экран осуществляется с помощью цикла DO WHILE...ENDDO. Команда ? выводит заголовок "Фамилия Имя Отчество Дата рождения", а команды DISPLAY OFF NEXT 5 - очередные пять записей. Затем пропускается пустая строка (команда ?), вывод записей приостанавливается и высвечивается сообщение "Для продолжения вывода нажмите любую клавишу. Для прекращения вывода - клавишу V". Вывод этого сообщения с одновременной приостановкой выполнения программы осуществляется командой WAIT, которая также вводит в переменную pv код нажатой клавиши, после чего происходит идентификация этой клавиши. При этом используется функция UPPER(), преобразующая строчную букву в прописную, поскольку в программе задан выход только по прописной букве. Тем самым пользователю предоставляется возможность нажимать для выхода как прописную, так и строчную букву v, которые имеют разные коды ASCII. Если нажата буква V, то командой EXIT осуществляется выход из цикла DO WHILE...ENDDO и выполняются команды, завершающие программу - закрытие фильтра и файла. Нажатие любой другой клавиши приводит к продвижению на одну запись указателя записей файла (командой SKIP) и повторению цикла операций. При этом реализуется проверка на конец файла, и если он обнаружен, то также выполняется выход из цикла и завершение программы.

```
* Output.prg
SET ECHO OFF
SET TALK OFF
SET DATE GERMAN

pdat1=CTOD("01.01.35")
pdat2=DATE()
@ 7,22 TO 15,52
@ 9,34 SAY "ВВЕДИТЕ" COLOR R*/N
@ 11,24 SAY "- начальную дату: " GET pdat1
@ 13,24 SAY "- конечную  дату: " GET pdat2
READ

```

```
USE BD1
SET FILTER TO d_r >= pdat1 .AND. d_r <= pdat2
GO TOP
SET FIELDS TO fam,nam,otc,d_r
CLEAR

DO WHILE .T.
  ? " Фамилия          Имя       "+;
    " Отчество     Дата рождения"
  DISPLAY  OFF NEXT 5
  ?
  WAIT "Для продолжения вывода нажать любую клавишу."+;
       "Для прекращения вывода - клавишу V" TO pv
  ?
  IF UPPER(pv) = 'V'
    EXIT
  ENDIF

  IF EOF()
    EXIT
  ELSE
    SKIP
  ENDIF
ENDDO

SET FILTER TO
USE
RETURN
```

Ввод и удаление данных в индексированных полях файла БД сопровождается автоматической переиндексацией записей. В результате чего на удаление неверных записей затрачивается время, пропорциональное емкости файла. Поэтому ввод данных в поля файла БД целесообразно выполнять не непосредственно, а через переменные, которые служат для промежуточного временного хранения. Данные, введенные в переменные, можно проверить и если они правильные, то ввести в файл БД, а если нет, то отказаться от ввода. Такая организация ввода уменьшает вероятность занесения в файл БД неверных данных.

Процедура **Input** организует ввод данных в файл БД через переменные. В начале процедуры выполняются команды установки состояний, подобно предыдущей процедуре. Команда SET DEVICE TO SCREEN устанавливает вывод сообщений на экран последующих выполняемых команд @...SAY.

Далее определяются переменные, в которые будет вводиться марка автомобиля (pmar), номерной знак (pn_z), состояние автомобиля (psos) и место его нахождения (pg_n) перед занесением в файл БД с именем BD2. Длина переменных и тип должны совпадать с длиной и типом полей файла БД. В переменную dn заносится значение "Да", которое будет использоваться для определения ответа пользователя.

Ввод данных в переменные осуществляется командами @... GET внутри цикла DO WHILE...ENDDO, который позволяет выполнить повторный ввод после ответа "Да" на вопрос "Редактировать данные ?". Ответ на этот вопрос можно ввести с клавиатуры, но лучше нажатием клавиши **Пробел**. Такой выбор реализуется с помощью использования функции @M (задание списка альтернативных значений переменной) в шаблоне ввода-вывода переменной dn.

Если дан ответ "Нет", то осуществляется выход из цикла, очищается 21-я строка экрана и выполняется проверка того, ввел ли пользователь данные хотя бы в одну переменную. Если данные не вводились, то командой RETURN выполняется выход из программы. В противном случае управление передается

на блок процедуры, который снова высвечивает введенные в переменные данные.

После команд @...GET, высвечивающих содержимое переменных, выполняется команда CLEAR GETS, которая деактивизирует эти команды и передает управление на следующую за ней команду @...GET. В последней команде запрашивается разрешение на ввод данных из переменных в поля файла БД. Если команду CLEAR GETS не поставить, все команды @...GET будут активизированы командой READ, тогда как требуется активизировать только команду запроса на ввод в БД. При ответе на этот запрос в переменную dn вводится значение "Да" или "Нет".

Затем анализируется значение переменной dn, и если оно равно "Да", то открывается файл БД и выполняется команда APPEND BLANK для ввода пустой записи в файл БД, а затем команда RERLACE по замещению пустых полей этой записи данными из переменных.

```
* Input.prg
CLEAR
SET ECHO OFF
SET TALK OFF
SET DEVICE TO SCREEN
pmar =SPACE(10)
pn_z =SPACE(9)
psos =SPACE(12)
pg_n =SPACE(12)
dn ="Да "
@ 6,19 TO 14,56
DO WHILE dn ="Да"
  @ 7,21 SAY "Марка автомобиля    " GET pmar
  @ 9,21 SAY "Номерной знак       " GET pn_z
  @ 11,21 SAY "Состояние автомобиля " GET psos
  @ 13,21 SAY "Где находится       " GET pg_n
  READ
  @ 21,20 SAY "Редактировать данные ? (Да/Нет) ";
       GET dn PICTURE "@M Да,Нет"
  READ
ENDDO
@ 21,0
IF pmar=' '.OR.pn_z=' '.OR.psos=' '.OR.pg_n=' '
  RETURN
ENDIF
@ 7,21 SAY "Марка автомобиля    " GET pmar
@ 9,21 SAY "Номерной знак       " GET pn_z
@ 11,21 SAY "Состояние автомобиля " GET psos
@ 13,21 SAY "Где находится       " GET pg_n
CLEAR GETS
@ 21,15 SAY "Сохранить данные в базе данных ? (Да/Нет) ";
       GET dn PICTURE '@M Да,Нет'
READ
IF dn ='Да '
  USE BD2
  APPEND BLANK
  REPLACE n_z WITH pn_z,sos WITH psos,;
```

```
        mar WITH pmar,g_n WITH pg_n
ENDIF

RETURN
```

Процедура **Key** иллюстрирует возможности назначения новых функций клавишам F1-F10. В этой процедуре функциональной клавише F3 назначается вызов процедуры с именем Spisfam, реализующей вывод и просмотр списка фамилий водителей из файла БД BD1. Предварительно инициируется символьная переменная pfam и определяется окно для высвечивания списка фамилий. Далее открывается файл БД BD1 и устанавливается главный (управляющий) индекс на поле фамилии fam. Затем командой @...SAY высвечивается предложение нажать клавишу F3 для вывода списка фамилий. После того как нужная фамилия найдена, ее можно ввести в переменную pfam с помощью команды @...GET и использовать для поиска данных, например в файле перевозок, выполненных этим водителем.

После нажатия клавиши F3 вызывается процедура Spisfam и на экране активизируется ранее определенное окно с именем Spisfam. В этом окне командой DISPLAY ALL высвечиваются фамилии, имена, отчества шести водителей. Следующая порция из этого списка выводится после нажатия любой клавиши (свойство команды DISPLAY). По достижении конца файла выполнение команды приостанавливается и высвечивается соответствующее сообщение. При нажатии любой клавиши командой DEACTIVATE WINDOW удаляется окно Spisfam и снова высвечивается сообщение о функции клавиши F3. Затем управление возвращается в строку, следующую после команды вызова процедуры Spisfam.

```
* Key.prg
SET ECHO OFF
SET TALK OFF
SET HEADING OFF

pfam=SPACE(15)
DEFINE WINDOW Spisfam FROM 10,12 TO 18,65 COLOR BG+/B

ON KEY LABEL F3 DO Spisfam

USE BD1
SET ORDER TO fam

@ 21,20 SAY "Просмотр списка водителей - клавиша F3"
@ 5,12 TO 7,65
@ 6,16 SAY "Введите фамилию водителя - " GET pfam
READ
* . . .
* . . .
* . . .
RETURN

PROCEDURE Spisfam
@ 21,0
ACTIVATE WINDOW Spisfam
DISPLAY ALL OFF fam,nam,otc
WAIT " ... Конец файла БД, нажмите любую клавишу "
DEACTIVATE WINDOW Spisfam
@ 21,20 SAY "Просмотр списка водителей - клавиша F3"
RETURN
```

Для пояснения функций управляющих клавиш и предоставления других данных о разработанной прикладной программе в ней обычно предусматривается

вывод справочной информации на экран нажатием одной из функциональных клавиш F1-F10 (обычно клавиши F1).

Процедура **Help.prg** иллюстрирует создание системы справочной информации о выполнении некоторых операций в стандартных экранных формах EDIT и BROWSE. Для вызова этой информации в процедуре командой DEFINE POPUP создается вертикальное меню с именем P1, а командой DEFINE WINDOW определяется окно с именем Hlp, в котором будет высвечиваться справочная информация.

Команда ON SELECTION определяет процедуру, которая будет выполняться при выборе позиций меню P1. Затем функциональной клавише F1 назначается выполнение процедуры Hlp, которой передается значение параметра phlp - число 1,2 или 3, определяющее соответственно высвечивание одного из трех экранов справочной информации. Первый экран - это информация о типах стандартных экранных форм Browse и Edit и меню P1, второй - информация об управляющих клавишах формы Browse, а третий - аналогичная информация о форме Edit.

Далее открывается файл базы данных BD1, данные из которого будут высвечиваться в стандартной форме Browse или Edit, и активизируется меню P1. В следующей строке закрывается файл БД и процедура завершается. Описанная часть представляет собой главную часть данной процедуры. Далее следуют вызываемые из нее процедуры.

В процедуре Proc_p1 для выполнения функций вертикального меню используется команда DO CASE...ENDCASE, в которой после опций CASE BAR() = i (где i - номер позиции вертикального меню P1) выполняется вывод данных в соответствующей стандартной экранной форме и присваивается требуемое значение переменной phlp, которое используется при вызове процедуры Hlp после нажатия функциональной клавиши F1. В процедуре Hlp сначала активизируется окно Hlp, затем в зависимости от значения переданного параметра phlp осуществляется ветвление управления в команде DO CASE...ENDCASE, после чего командой TEXT...ENDTEXT высвечивается текст справочной информации. Этот текст удерживается на экране благодаря приостановке выполнения программы последующей командой xx = INKEY(0). Если весь текст не умещается на одном экране, он разбивается на несколько частей. Ниже показан экран N 3 справочной информации.

```
Records    Fields    Go To    Exit                         10:28:48 am
┌───┬──────────────────┬──────────────────┬──────────────────┬──────┬───┬─────┬───┐
│NOM│FAM               │NAM               │OTC               │D_R   │S_R│KAT  │AU │
╞═══╪══════════════════╪══════════════════╪══════════════════╪══════╪═══╪═════╪═══╡
│  1│Арх┌────────────────────────────────────────────────────────────────┐   │ME │
│  2│Ста│Экран помощи N3  Основные функциональные клавиши                │   │ME │
│  3│Без│---------------             в форме BROWSE                      │,C,│ME │
│  4│Бор│                                                                │   │ME │
│  5│Ерм│F1          - вызов экрана помощи                               │,A │ME │
│  6│Мал│F2          - переключение между Edit и Browse формой           │,C │ME │
│  7│Пол│F10         - активизация меню                                  │,E │ME │
│  8│Ков│SHIFT-F2    - переход в экран создания запроса                  │   │ME │
│  9│Сад│SHIFT-F8    - копирование содержимого поля из                   │   │ME │
│ 10│Стр│              предыдущей записи в текущую                       │   │ME │
│ 11│Бро│SHIFT-F9    - распечатка содержимого файла БД                   │   │ME │
│   │   │TAB   - движение по полям вправо                                │   │   │
│   │   │SHIFT-TAB - движение по полям влево                             │   │   │
│   │   │Для продолжения просмотра нажмите любую клавишу ...             │   │   │
│   │   └────────────────────────────────────────────────────────────────┘   │   │
└───┴──────────────────┴──────────────────┴──────────────────┴──────┴───┴─────┴───┘
  Browse  │D:\dbase\cod\BD1 │Rec 1/11 │File │ View and edit fields
```

```
* Help.prg
CLEAR
SET ECHO OFF
SET TALK OFF
PUBLIC phlp
DEFINE WINDOW Hlp FROM 4,9 TO 18,69 NONE COLOR BG+/B

DEFINE POPUP P1 FROM 7,25
DEFINE BAR 1 OF P1 PROMPT "        E D I T          "
DEFINE BAR 2 OF P1 PROMPT "──────────────────────────" SKIP
DEFINE BAR 3 OF P1 PROMPT "      B R O W S E        "
DEFINE BAR 4 OF P1 PROMPT "──────────────────────────" SKIP
DEFINE BAR 5 OF P1 PROMPT "  Выход из программы     "
ON SELECTION POPUP P1 DO Proc_p1

ON KEY LABEL F1 DO Hlp WITH phlp

USE BD1
phlp=1
ACTIVATE POPUP P1
USE
RETURN

PROCEDURE Proc_p1
DO CASE
   CASE BAR()=1
   phlp=2
   EDIT
   CASE BAR()=3
   phlp=3
   BROWSE
   CASE BAR()=5
   DEACTIVATE POPUP
ENDCASE
phlp=1
RETURN

PROCEDURE Hlp
PARAMETER phlp
ACTIVATE WINDOW Hlp
DO CASE
   CASE phlp=1
   TEXT
   Экран помощи N1       Функции вертикального меню
   -------------------------
   ┌─────────────────────────────────┐
   │Формы представления данных       │  E D I T - высвечивает на
   │━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━│  экране одновременно одну
   │         E D I T                 │  запись файла БД. Поля
   │─────────────────────────────────│  размещаются одно под
   │       B R O W S E               │  другим в столбик.
   │─────────────────────────────────│
   │   Выход из программы            │  B R O W S E - выводит
   └─────────────────────────────────┘  данные из файла БД в
   табличном виде. Строка таблицы - запись, колонка - поле.

   Выход из программы - завершение работы с программой.
   ENDTEXT
   CASE phlp=2
```

```
   TEXT
   Экран помощи N2   Основные функциональные клавиши
   --------------              в форме EDIT

   F1        - вызов экрана помощи
   F2        - переключение между Edit и Browse формой
   F10       - активизация меню
   SHIFT-F2  - переход в экран создания запроса
   SHIFT-F8  - копирование содержимого поля из предыдущей
               записи в текущую
   SHIFT-F9  - распечатка содержимого файла БД
   TAB       - движение по полям вниз
   SHIFT-TAB - движение по полям вверх

   Продолжение - любая клавиша
   ENDTEXT
   xx=INKEY(0)
   clear
   TEXT
   HOME      - переход к первому полю
   END       - переход к последнему полю записи
   CTRL-END  - выход с сохранением
   ESC       - выход без сохранения

   Для выхода из экрана помощи нажмите любую клавишу
   ENDTEXT

   CASE phlp=3
   TEXT
   Экран помощи N3   Основные функциональные клавиши
   --------------              в форме BROWSE

   F1        - вызов экрана помощи
   F2        - переключение между Edit и Browse формой
   F10       - активизация меню
   SHIFT-F2  - переход в экран создания запроса
   SHIFT-F8  - копирование содержимого поля из предыдущей
               записи в текущую
   SHIFT-F9  - распечатка содержимого файла БД
   TAB       - движение по полям вправо
   SHIFT-TAB - движение по полям влево

   Для продолжения просмотра нажмите любую клавишу
   ENDTEXT
   xx=INKEY(0)
   CLEAR
   TEXT
   HOME      - переход к первому полю
   END       - переход к последнему полю записи
   PGDN      - переход по строкам и страницам
   PGUP        таблицы
   CTRL-END  - выход с сохранением
   ESC       - выход без сохранения

   Для выхода из экрана помощи нажмите любую клавишу
   ENDTEXT
ENDCASE
xx=INKEY(0)
DEACTIVATE WINDOW Hlp
RETURN
```

В прикладных программах при выводе на печать часто приходится устанавливать параметры печати. Эти действия можно оформить в виде отдельной процедуры. Проиллюстрируем, как создать подобную процедуру на примере процедуры **Mnuprint**, позволяющей назначить параметры, которые высвечиваются на экране, как показано ниже.

```
 Формат страницы                        Параметры печати
--------------------------------------------------------------------

 Отступ слева             0   Начать печать со страницы N    1
 Левая граница            0   Закончить на странице N      9999
 Правая граница          80   Число копий                     1
 Абзац                    4   Прогон бумаги                NONE
 Колич. строк на стр.    66   Ждать между страницами       N
 Расст. между строками    1   Тип шрифта                   PICA
                              Качественная печать          N

            Для выхода  нажмите клавишу   PgDn
```

Установка параметров осуществляется посредством ввода значений в переменные, которые в дальнейшем присваиваются соответствующим системным переменным.

В начале программы определяется окно, в котором будут высвечиваться параметры. Затем переменным присваиваются начальные значения, активизируется окно Mprint, в котором командами @...SAY и @...GET высвечиваются наименования и начальные значения параметров. При установке курсора на поле в нижней строке окна с помощью опции MESSAGE команд @...GET высвечиваются сообщения о типе вводимого значения. В нижней строке окна командой @...SAY выводится информация для пользователя, как выйти из процедуры после ввода требуемых изменений или принятия высвеченных значений.

Нажатие клавиши PgDn означает переход к группе команд @...SAY и @...GET, располагаемых на следующей странице экрана, или при высвечивании только одной страницы - переход к следующей за командами @...SAY и @...GET команде READ, которая записывает содержимое GET-полей в соответствующие переменные. Затем командой DEACTIVATE WINDOW деактивизируется окно Mprint и системным переменным присваиваются значения из соответствующих переменных процедуры. Для удобства распознавания переменных в процедуре им присваиваются те же имена, что и системным переменным, но без ведущих символов.

```
* Mnuprint.prg
CLEAR
SET ECHO OFF
SET TALK OFF
SET STATUS OFF

DEFINE WINDOW Mprint FROM 5,2 TO 17,79 DOUBLE COLOR BG+/b

msg_num   = "Введите число"
msg_logic = "Введите Y или N"
msg_enum  = "Для выбора опций нажмите клавишу <Пробел>"
loffset  = 0
lmargin  = 0
rmargin  = 80
indent   = 4
```

```
plength  = 66
STORE 1 TO pspacing, pbpage, pcopies
pepage   = 9999
peject   = "NONE  "
STORE .F. TO pwait, pquality
ppitch   = "PICA      "
ACTIVATE WINDOW Mprint
@ 1,0  SAY REPLICATE ('═',76)
@ 0,1  SAY "       Формат страницы        "
@ 3,1  SAY " Отступ слева                 ";
       GET loffset PICTURE "99" MESSAGE msg_num
@ 4,1  SAY " Левая граница                ";
       GET lmargin PICTURE "99" MESSAGE msg_num
@ 5,1  SAY " Правая граница               ";
       GET rmargin PICTURE "999" MESSAGE msg_num
@ 6,1  SAY " Абзац                        ";
       GET indent PICTURE "99" MESSAGE msg_num
@ 7,1  SAY " Количество строк на странице ";
       GET plength PICTURE "99" MESSAGE msg_num
@ 8,1  SAY " Расстояние между строками    ";
       GET pspacing PICTURE "9" RANGE 1,3 MESSAGE msg_num
@ 0,36 SAY "       Параметры печати       "
@ 3,36 SAY " Начать печать со страницы N ";
       GET pbpage PICTURE "999"  MESSAGE msg_num
@ 4,36 SAY " Закончить на странице N     ";
       GET pepage   PICTURE "9999" MESSAGE msg_num
@ 5,36 SAY " Число копий                 ";
       GET pcopies  PICTURE "999"  MESSAGE msg_num
@ 6,36 SAY " Прогон бумаги               ";
       GET peject PICTURE "@M BEFORE,AFTER,BOTH,NONE";
       MESSAGE msg_enum
@ 7,36 SAY " Ждать между страницами      ";
       GET pwait   PICTURE "Y" MESSAGE msg_logic
@ 8,36 SAY " Тип шрифта                  ";
       GET ppitch PICTURE "@M DEFAULT,PICA,ELITE,CONDENSED";
       MESSAGE msg_enum
@ 9,36 SAY " Качественная печать         ";
       GET pquality PICTURE "Y" MESSAGE msg_logic
READ
DEACTIVATE WINDOW Mprint
_ploffset = loffset
_lmargin  = lmargin
_rmargin  = rmargin
_indent   = indent
_plength  = plength
_pspacing = pspacing
_pbpage   = pbpage
_pepage   = pepage
_pcopies  = pcopies
_peject   = peject
_pwait    = pwait
_ppitch   = ppitch
_pquality = pquality
RETURN
```

Важной проблемой при разработке прикладных программ является обработка ошибочных ситуаций, которые могут возникнуть при их отладке и последующей эксплуатации. Такая обработка предполагает вывод как минимум информации о месте, в котором возникла такая ситуация. Часто можно также вывести информацию о характере ошибки и предложение выполнить действия, устраняющие ее. В случае неустранимости ошибочной ситуации целесообразно передать управление на завершение программы. Возможности по обработке ошибочных ситуаций можно продемонстрировать на примере процедуры **Error.prg**, в которой обрабатываются три имитируемые ошибочные ситуации: вывод данных на принтер при его неготовности, поиск данных в неиндексированном поле командой SEEK, которая используется только для индексированных полей, активизация несуществующего вертикального меню.

В начале программы командой DEFINE WINDOW определяется окно Werr, в котором будут высвечиваться сообщения об ошибках. Затем командой ON ERROR устанавливается состояние, при котором ошибочная ситуация приводит к вызову процедуры Oshibka, обрабатывающей ошибочную ситуацию. Вместе с ней передается значение параметра, содержащее имя процедуры и номер строки, в которой возникла ошибка. Затем вызывается процедура SITUAC для имитации указанных выше ошибочных ситуаций. После выхода из ошибочной ситуации состояние ON ERROR выключается и главная процедура завершается.

Дальше следуют тексты вызываемых процедур. В начале процедуры Oshibka переменной perr присваивается текущее значение функции ERROR() - номер ошибки. Затем активизируется окно Werr для высвечивания сообщения об ошибке и командой @...SAY в начале третьей строки окна выводится слово "Сообщение", общее для всех сообщений об ошибках. Далее выполняется команда DO CASE...ENDCASE, в которой управление разветвляется в зависимости от переданного в параметре perr номера ошибки и высвечиваются соответствующие сообщения. В начале каждого сообщения выводится имя процедуры и номер строки, в которой произошла ошибка. Эти данные передаются в процедуру как значение параметра pmes. Высвечиваемые командами сообщения удерживаются на экране командой WAIT.

Для продолжения выполнения программы нужно нажать любую клавишу. Сообщение об этом выводится командой SET MESSAGE в нижней строке экрана. Команда SET CONSOLE OFF подавляет вывод на экран любой информации, в том числе системного сообщения команды WAIT, (кроме информации, выводимой командами @...SAY, системных сообщений об ошибках и предупреждений о перезаписи и удалении файлов). При нажатии любой клавиши после приостановки выполнения командой WAIT, команда SET CONSOLE ON восстанавливает вывод на экран системной информации, а команда SET MESSAGE TO удаляет высвеченное ранее сообщение в нижней строке экрана. Затем деактивизируется окно Werr и управление возвращается в главную процедуру. Ниже показано окно при выводе сообщения об ошибке, возникшей при выводе на принтер.

```
Ошибка в процедуре  SITUAC.      Строка N: 52

Сообщение:  Принтер не готов !
            Включите принтер (если он не включен)
            и переведите его в состояние ON LINE.
```

В процедуре Situac командой SET DEVICE TO PRINTER устанавливается вывод сообщений команд @...SAY на принтер, что при неготовности принтера

(не включенном в сеть или находящемся в состоянии OFF LINE) вызывает ошибку с номером 125. После обработки ошибочной ситуации и установки принтера в состояние ON LINE управление передается на команду SET DEVICE TO SCREEN, которая восстанавливает вывод сообщений команд @...SAY на экран.

Для получения ошибки с номером 26 открывается файл БД BD1 и не устанавливается управляющий индекс. Выполнение команды SEEK поиска данных по неиндексированному полю приводит к ошибочной ситуации. После обработки этой ситуации командой USE выполняется закрытие файла БД.

Команда ACTIVATE POPUP Pp вызывает ошибку с номером 165, если вертикального меню с именем Pp в текущем директории не существует.

```
* Error.prg
CLEAR
SET ECHO OFF
SET TALK OFF
DEFINE WINDOW Werr FROM 6,13 TO 15,66  COLOR GR+/R
ON ERROR DO Oshibka WITH 'Ошибка в процедуре  '+PROGRAM()+;
'.      Строка N: '+LTRIM(STR(LINE()))
DO Situac
ON ERROR
RETURN
PROCEDURE Oshibka
PARAMETER pmes
perr =ERROR()
ACTIVATE WINDOW werr
@ 3,2 SAY "Сообщение:" color w+/r
DO CASE
   CASE perr = 26
   @ 1,2 SAY pmes
   @ 3,14 SAY "Файл БД не проиндексирован !        "
   @ 4,14 SAY "Откройте индексный файл.           "
   . . .
   . . .
   . . .
   CASE perr = 125
   @ 1,2  SAY pmes
   @ 3,14 SAY "Принтер не готов !          "
   @ 4,14 SAY "Включите принтер (если он не включен)"
   @ 5,14 SAY "и переведите его в состояние ON LINE."
   CASE perr = 165
   @ 1,2  SAY pmes
   @ 3,14 SAY "Вертикальное меню не определено !  "
   @ 4,14 SAY "Вы  должны  определить  меню  перед  "
   @ 5,14 SAY "активизацией  или  определением  его  "
   @ 6,14 SAY "позиций. "
ENDCASE
SET MESSAGE TO "... для продолжения нажмите любую клавишу"
SET CONSOLE OFF
WAIT
SET CONSOLE ON
SET MESSAGE TO
```

```
DEACTIVATE WINDOW Werr
RETURN

PROCEDURE Situac
SET DEVICE TO PRINTER
SET DEVICE TO SCREEN

USE BD1
SEEK '1'
USE

ACTIVATE POPUP Pp
RETURN
```

Ввод и обновление данных в файлах БД является важной составной частью эксплуатации прикладной программы. В гл. 2 рассматривалась задача учета перевозок автотранспортным предприятием.

Данные в этой задаче представлены в виде четырех связанных файлов БД. Ключевой из них - файл BD1, содержащий сведения о водителях (фамилию, имя, отчество, номерной знак автомобиля, на котором работает и т.д.). Данные о выполненных водителями перевозках содержатся в файле BD3, который связан с файлом BD1 по общему полю номерного знака.

При вводе данных о перевозках нужно сначала найти фамилию водителя в файле BD1, затем перейти в файл BD3, по номерному знаку его автомобиля найти ранее введенные записи о перевозках и добавить новую запись.

Процедура **Append** реализует эти операции по вводу новых записей о перевозках. В верхней части экранной формы этой процедуры, показанной ниже, высвечиваются данные из файла БД BD1, содержащего сведения о водителях, а в нижней - данные из файла БД перевозок BD3. Пользователь, выбрав фамилию, нажимает клавишу Esc и переходит к заполнению информации по перевозкам грузов, выполненных данным водителем.

8:20:40pm

09-33 ММД	Бронзов	Сергей	Васильевич
01-34 ММД	Старченко	Сергей	Николаевич
34-56 ММД	Строков	Олег	Валерьевич

Груз	Вес	Откуда	Куда	Выезд	Прибытие	Время	Горючее
Станки ЧПУ	3.0	Одинцово	Горький	24/03/89	25/03/89	14	44.00
Мебель	1.0	Москва	Калуга	30/06/89	01/07/89	12	46.00
Двигатели	3.0	Москва	Рязань	12/10/89	14/10/89	22	72.00

Переход к водителям - <CTRL-END>. Выход из программы - F8

Процедура начинается с определения двух окон W1 и W2 для высвечивания данных из файлов БД BD1 и BD3. Затем клавише F8 назначается функция завершения процедуры командой RETURN. Команда SELECT 1 активизирует рабочую область 1, в которой последующей командой открывается файл БД BD1 с управляющим индексом в поле fam. Этому файлу назначается алиас A, который в дальнейшем можно использовать для обращения к его полям, если рабочая область не активна. Командой SELECT 2 активизируется рабочая область 2, в которой открывается файл БД BD3 и ему назначается алиас B. Следующей командой SELECT 1 выбирается (активизируется) рабочая область 1 и открывается цикл DO WHILE...ENDDO, в котором и выполняются все операции по вводу данных о перевозках.

В этом цикле сначала в 22-й строке экрана командой @...SAY высвечивается текст сообщения с предложением выбрать фамилию и функции клавиш Esc и F8. Затем выводятся данные из файла BD1 в окне W1 в стандартной форме Browse. В команде BROWSE ширина выводимого поля n_z ограничивается 22 позициями и все поля, кроме поля fam, опцией FREEZE закрываются для редактирования (курсор не может перемещаться в эти поля).

После выбора фамилии и выхода из формы Browse командой @ 22,0 очищается 22-я строка экрана и переменной vn_z присваивается значение поля n_z файла BD1 (впереди поля стоит алиас этого файла). Заметим, что алиас в данном случае ставить не обязательно, так как в текущий момент активизирована рабочая область именно этого файла. Затем выбирается рабочая область 2 и в ней устанавливается фильтр на отбор записей с номерным знаком, взятым из переменной vn_z. После отбора этих записей командой APPEND BLANK в конец добавляется пустая запись и затем командой REPLACE значение поля n_z замещается значением переменной vn_z. Следующая за ней команда SET CARRY TO n_z обеспечивает автоматический перенос значения поля n_z из предыдущей записи в новую добавляемую запись.

Затем командой @...SAY высвечивается сообщение с именами клавиш, используемых для перехода из нижней части экрана (файл перевозок) в верхнюю часть экрана (файл данных о водителях) выводится шапка таблицы Browse файла BD3 и выполняется собственно команда вывода данных в эту таблицу. После выхода из таблицы Browse клавишами Ctrl-End очищаются строки экрана (10,11 и 22), снимается фильтр, выбирается рабочая область 1 и осуществляется переход на следующую запись в файле BD1.

```
* Append.prg
CLEAR
SET BELL OFF
SET TALK OFF
SET ECHO OFF
SET STATUS OFF

DEFINE WINDOW W1 FROM  1,0 TO  5,79 NONE COLOR W/B,GR+/R,B/B
DEFINE WINDOW W2 FROM 12,0 TO 20,79 NONE COLOR N/BG,GR+/R,BG/BG
ON KEY LABEL F8 RETURN

SELECT 1
USE BD1 ORDER fam ALIAS A
SELECT 2
USE BD3 ALIAS B

SELECT 1
DO WHILE .T.

 @ 22,1 SAY "Выберите  фамилию и нажмите клавишу"+;
           " <ESC>.  Выход из программы - F8" COLOR G+/N
```

```
BROWSE FIELDS n_z/22,fam,nam,otc COMPRESS NOCLEAR;
     WINDOW W1 FREEZE fam
@ 22,0
vn_z=A->n_z

SELECT 2
SET FILTER TO n_z=vn_z
APPEND BLANK
REPLACE n_z WITH vn_z
SET CARRY TO n_z

@ 22,8 SAY " Переход к данным о водителях - <CTRL-END>."+;
         " Выход из программы - F8 " COLOR G+/N
@ 10,0 SAY "    Груз      Вес  Откуда   Куда "+;
         "  Выезд  Прибытие Время  Горю-   "
@ 11,59 SAY "     чее        "
BROWSE FIELDS n_g/14,ves,o_p/8,k_d/9,d_v,d_p,vrm/4,rgr,n_z;
     COMPRESS WINDOW W2 NOMENU
@ 10,0
@ 11,0
@ 22,0
SET FILTER TO
SELECT 1
SKIP

ENDDO
RETURN
```

Эксплуатация прикладных программ пользователем обычно осуществляется с помощью систем меню, которые можно представить в различном виде. Однако практика показывает, что наиболее употребительными являются меню горизонтального и вертикального типов и их комбинации. Своего рода стандартом стала форма меню, состоящая из главного горизонтального меню, к позициям которого прикрепляются вертикальные меню нескольких уровней. Создание такого типа меню иллюстрирует процедура **Menu**, в которой горизонтальное меню состоит из трех позиций, размещаемых в верхней части экрана. К первым двум позициям прикрепляются вертикальные меню. Вертикальное меню, прикрепленное к первой позиции, - одноуровневое, а ко второй позиции - двухуровневое. Третья позиция горизонтального меню является исполнимой и в ней нет уровней. Эта система меню показана ниже.

```
 Водители                        Автомобили          Выход
+------------------------------+ +-----------+
| Стаж работы - больше 5 лет   | | В автопарке |
|------------------------------| |-----------|
| Категории А,В,С              | | В рейсе   +----------------------+
+------------------------------+ |-----------| На заводе            |
                                 | В ремонте |----------------------|
                                 +-----------| На автопредприятии   |
                                             +----------------------+
```

Процедура начинается с команды чистки экрана CLEAR. Затем командой DEFINE MENU определяется горизонтальное меню с именем Gm и командами DEFINE PAD тексты его позиций с именами P1, P2 и P3. Команда ON PAD обеспечивает активизацию вертикального меню с именем Pp1 после выбора позиции P1 горизонтального меню. Команда ON SELECTION обеспечивает вызов процедуры Proc2 после выбора позиции P2 горизонтального меню. В этой процедуре активизируется вертикальное меню второго уровня. Следующая

команда ON SELECTION назначает выполнение команды RETURN при выборе позиции P3 горизонтального меню.

Дальше размещаются команды, определяющие вертикальные меню Pp1, Pp2 и Pp3. Тексты позиций этих меню назначаются командами DEFINE BAR i, где i - номер позиции. После команд определения текста позиций следует команда ON SELECTION POPUP, которая устанавливает вызов процедур Proc_pp1, Proc_pp2 и Proc_pp3, реализующих функции соответствующих вертикальных меню. Затем командой ACTIVATE MENU активизируется горизонтальное меню Gm и командой RETURN завершается главная процедура, ниже которой размещаются процедуры, вызываемые при выборе позиций меню. В этих процедурах используется команда DO CASE...ENDCASE для разветвления управления в зависимости от номера выбранной активной позиции меню. Этот номер передается с помощью функции BAR(). Неактивные позиции, в которых задается опция SKIP, служат для рисования отделяющих горизонтальных линий, информационного текста и т.д.

```
* Menu.prg
CLEAR
DEFINE MENU Gm
DEFINE PAD P1 OF Gm PROMPT " Водители   " AT 0,1
DEFINE PAD P2 OF Gm PROMPT " Автомобили " AT 0,20
DEFINE PAD P3 OF Gm PROMPT " Выход "     AT 0,40

ON PAD P1 OF GM ACTIVATE POPUP Pp1
ON SELECTION PAD P2 OF Gm ACTIVATE POPUP Pp2
ON SELECTION PAD P3 OF Gm RETURN

DEFINE POPUP Pp1 FROM 3,1
DEFINE BAR 1 OF Pp1 PROMPT " Стаж работы - больше 5 лет"
DEFINE BAR 2 OF Pp1 PROMPT "————————————————————————————" SKIP
DEFINE BAR 3 OF Pp1 PROMPT " Категории A,B,C            "
ON SELECTION POPUP Pp1 DO Proc_pp1

DEFINE POPUP Pp2 FROM 3,21
DEFINE BAR 1 OF Pp2 PROMPT " В автопарке   "
DEFINE BAR 2 OF Pp2 PROMPT "———————————————" SKIP
DEFINE BAR 3 OF Pp2 PROMPT " В рейсе       "
DEFINE BAR 4 OF Pp2 PROMPT "———————————————" SKIP
DEFINE BAR 5 OF Pp2 PROMPT " В ремонте     "
ON SELECTION POPUP Pp2 DO Proc_pp2

DEFINE POPUP Pp23 FROM 6,34
DEFINE BAR 1 OF Pp23 PROMPT " На заводе         "
DEFINE BAR 2 OF Pp23 PROMPT "———————————————————" SKIP
DEFINE BAR 3 OF Pp23 PROMPT " На автопредприятии "
ON SELECTION POPUP Pp23 DO Proc_pp23

ACTIVATE MENU Gm

RETURN


PROCEDURE Proc_pp1
DO CASE
  CASE BAR()=1
  * . . .
  CASE BAR()=3
```

```
 * ...
ENDCASE

RETURN

PROCEDURE Proc_pp2
DO CASE
 CASE BAR()=1
 * ...
 CASE BAR()=3
 * ...
 CASE BAR()=5
 ACTIVATE POPUP Pp23
ENDCASE
RETURN

PROCEDURE Proc_pp23
DO CASE
 CASE BAR()=1
 * ...
 CASE BAR()=3
 * ...
ENDCASE
RETURN

PROCEDURE Proc2
ACTIVATE POPUP Pp2
RETURN
```

В следующей процедуре **Music** демонстрируются возможности по озвучиванию прикладных программ. Озвучивание выполнения различных операций в прикладной программе создает удобство при эксплуатации.

В процедуре Music моделируется звучание нот в диапазоне от "до" до "си" при нажатии цифровых клавиш верхнего ряда клавиатуры от 1 до знака " = ", которым на экране, показанном ниже, соответствуют клавиши музыкального инструмента.

При нажатии клавиши клавиатуры компьютера красным цветом на экране подсвечивается соответствующая клавиша музыкального инструмента.



Процедура Music начинается с определения окна с именем Keyb, в котором будут высвечиваться клавиши музыкального инструмента. Затем командой @...SAY в 20-й строке экрана выводится сообщение о клавише, управляющей выходом из процедуры. Далее активизируется окно Keyb, в котором высвечиваются клавиши музыкального инструмента. Моделирование

звучания клавиш выполняется в цикле DO WHILE...ENDDO. В этом цикле сначала командой SET CONSOLE OFF подавляется вывод на экран сообщения, которое высвечивается при выполнении последующей команды WAIT. Эта команда приостанавливает выполнение программы до нажатия любой клавиши. Последняя нажатая клавиша анализируется с помощью функции LASTKEY() в главной процедуре. Если это "Пробел", то осуществляется выход из цикла, после чего деактивизируется окно Keyb, очищается 20-я строка экрана, и командой SET BELL TO восстанавливается частота и длительность звукового сигнала, принятые по умолчанию, и главная процедура завершается командой RETURN. В противном случае вызывается процедура Melod. После возвращения из этой процедуры продолжается выполнение цикла DO WHILE...ENDDO.

В процедуре Melod в соответствии с нажатой клавишей верхнего ряда клавиатуры моделируется звучание определенной ноты. После этого и в случае, если нажата клавиша, отличная от упомянутых клавиш верхнего ряда клавиатуры, управление возвращается в главную процедуру.

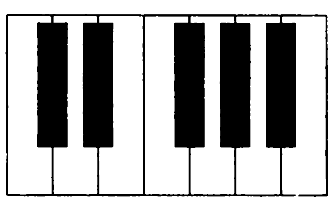
Процедура Melod содержит команду DO CASE...ENDCASE, в которой управление разветвляется в зависимости от нажатой клавиши, возвращаемой функцией LASTKEY(). После опции CASE переменным i и k присваиваются частота звучания и номер колонки на экране, в которой расположена левая сторона клавиши музыкального инструмента. Эти клавиши разделяются на четыре типа: левая, средняя, правая и диез/бемоль (черного цвета). Левая клавиша расположена слева от клавиши диез/бемоль, правая - справа от такой клавиши, а средняя - между этими клавишами. В зависимости от типа клавиши вызываются процедуры Klev0, Ksred0, Kprav0 и Dibemol0 для подсветки красным цветом соответствующей клавиши, и процедуры Klev1, Ksred1, Kprav1 и Dibemol1 для восстановления первоначального цвета.

Процедура Zvuk вызывает звуковой сигнал с частотой, соответствующей нажатой клавише, передаваемой в параметре i. Эта частота устанавливается командой SET BELL.

```
* Music.prg
SET ECHO OFF
SET TALK OFF
SET STATUS OFF
SET ESCAPE ON

DEFINE WINDOW Keyb FROM 4,23 TO 14,53 NONE COLOR G/G

@ 20,19 SAY "Для завершения нажмите клавишу <Пробел>"
ACTIVATE WINDOW Keyb
```



```
@ 1,1 SAY "   " COLOR BG+/B
@ 2,1 SAY "   " COLOR BG+/B
@ 3,1 SAY "   " COLOR BG+/B
@ 4,1 SAY "   " COLOR BG+/B
@ 5,1 SAY "   " COLOR BG+/B
@ 6,1 SAY "   " COLOR BG+/B
@ 7,1 SAY "   " COLOR BG+/B
@ 8,1 SAY "   " COLOR BG+/B
@ 9,1 SAY "   " COLOR BG+/B

DO WHILE .T.
  SET CONSOLE OFF
  WAIT
  SET CONSOLE ON

  IF LASTKEY() = 32
```

```
    EXIT
  ENDIF
  DO Melod

ENDDO

DEACTIVATE WINDOW Keyb
@ 20,0
SET BELL TO
RETURN

PROCEDURE Melod
DO CASE
  CASE LASTKEY() = 49          && "1"
    i =523
    k =2
    DO Klev0 WITH k
    DO Zvuk WITH i
    DO Klev1 WITH k
  CASE LASTKEY() = 50          && "2"
    i =552
    k =4
    DO Dibemol0 WITH k
    DO Zvuk WITH i
    DO Dibemol1 WITH k
  CASE LASTKEY() = 51          && "3"
    i =587
    k =7
    DO Ksred0 WITH k
    DO Zvuk WITH i
    DO Ksred1 WITH k
  CASE LASTKEY() = 52          && "4"
    i =612
    k =8
    DO Dibemol0 WITH k
    DO Zvuk WITH i
    DO Dibemol1 WITH k
  CASE LASTKEY() = 53          && "5"
    i =659
    k =11
    DO Kprav0 WITH k
    DO Zvuk WITH i
    DO Kprav1 WITH k
  CASE LASTKEY() = 54          && "6"
    i =698
    k =14
    DO Klev0 WITH k
    DO Zvuk WITH i
    DO Klev1 WITH k
  CASE LASTKEY() = 55          && "7"
    i =743
    k =16
    DO Dibemol0 WITH k
    DO Zvuk WITH i
    DO Dibemol1 WITH k
  CASE LASTKEY() = 56          && "8"
    i =784
    k =19
    DO Ksred0 WITH k
```

```
    DO Zvuk WITH i
    DO Ksred1 WITH k
  CASE LASTKEY() = 57          && "9"
    i=832
    k=20
    DO Dibemol0 WITH k
    DO Zvuk WITH i
    DO Dibemol1 WITH k
  CASE LASTKEY() = 48          && "0"
    i=880
    k=23
    DO Ksred0 WITH k
    DO Zvuk WITH i
    DO Ksred1 WITH k
  CASE LASTKEY() = 45          && "-"
    i=931
    k=24
    DO Dibemol0 WITH k
    DO Zvuk WITH i
    DO Dibemol1 WITH k
  CASE LASTKEY() = 61          && "="
    i=980
    k=27
    DO Kprav0 WITH k
    DO Zvuk WITH i
    DO Kprav1 WITH k
ENDCASE
RETURN

PROCEDURE Zvuk
PARAMETER i
SET BELL TO i,1
? CHR(7)
RETURN

PROCEDURE Klev0
PARAMETER k
@ 2,k SAY " " COLOR GR+/R
@ 3,k SAY " " COLOR GR+/R
@ 4,k SAY " " COLOR GR+/R
@ 5,k SAY " " COLOR GR+/R
@ 6,k SAY " " COLOR GR+/R
@ 7,k SAY "  " COLOR GR+/R
@ 8,k SAY "  " COLOR GR+/R
RETURN

PROCEDURE Klev1
PARAMETER k
@ 2,k SAY " " COLOR BG+/B
@ 3,k SAY " " COLOR BG+/B
@ 4,k SAY " " COLOR BG+/B
@ 5,k SAY " " COLOR BG+/B
@ 6,k SAY " " COLOR BG+/B
@ 7,k SAY "  " COLOR BG+/B
@ 8,k SAY "  " COLOR BG+/B
RETURN

PROCEDURE Ksred0
PARAMETER k
```

```
@ 2,k   SAY "  " COLOR GR+/R
@ 3,k   SAY "  " COLOR GR+/R
@ 4,k   SAY "  " COLOR GR+/R
@ 5,k   SAY "  " COLOR GR+/R
@ 6,k   SAY "  " COLOR GR+/R
@ 7,k-1 SAY "    " COLOR GR+/R
@ 8,k-1 SAY "    " COLOR GR+/R
RETURN

PROCEDURE Ksred1
PARAMETER k
@ 2,k   SAY "  " COLOR BG+/B
@ 3,k   SAY "  " COLOR BG+/B
@ 4,k   SAY "  " COLOR BG+/B
@ 5,k   SAY "  " COLOR BG+/B
@ 6,k   SAY "  " COLOR BG+/B
@ 7,k-1 SAY "    " COLOR BG+/B
@ 8,k-1 SAY "    " COLOR BG+/B
RETURN

PROCEDURE Kprav0
PARAMETER k
@ 2,k   SAY "   " COLOR GR+/R
@ 3,k   SAY "   " COLOR GR+/R
@ 4,k   SAY "   " COLOR GR+/R
@ 5,k   SAY "   " COLOR GR+/R
@ 6,k   SAY "   " COLOR GR+/R
@ 7,k-1 SAY "    " COLOR GR+/R
@ 8,k-1 SAY "    " COLOR GR+/R
RETURN

PROCEDURE Kprav1
PARAMETER k
@ 2,k   SAY "   " COLOR BG+/B
@ 3,k   SAY "   " COLOR BG+/B
@ 4,k   SAY "   " COLOR BG+/B
@ 5,k   SAY "   " COLOR BG+/B
@ 6,k   SAY "   " COLOR BG+/B
@ 7,k-1 SAY "    " COLOR BG+/B
@ 8,k-1 SAY "    " COLOR BG+/B
RETURN

PROCEDURE Dibemol0
PARAMETER k
@ 2,K SAY "███" COLOR GR+/R
@ 3,K SAY "███" COLOR GR+/R
@ 4,K SAY "███" COLOR GR+/R
@ 5,K SAY "███" COLOR GR+/R
@ 6,K SAY "███" COLOR GR+/R
RETURN

PROCEDURE Dibemol1
PARAMETER k
@ 2,K SAY "███" COLOR BG+
@ 3,K SAY "███" COLOR BG+
@ 4,K SAY "███" COLOR BG+
@ 5,K SAY "███" COLOR BG+
@ 6,K SAY "███" COLOR BG+
RETURN
```

5.3. ПРИКЛАДНАЯ ПРОГРАММА РЕАЛИЗАЦИИ СЕРВИСНЫХ ФУНКЦИЙ ВЕДЕНИЯ БАЗ ДАННЫХ

Прикладная программа Service позволяет выполнять ряд сервисных функций, подобных управляющему центру dBASE IV. Кроме этого, исходя из опыта работы с пакетом dBASE IV, в данной программе реализованы некоторые полезные функции, которые нельзя выполнить непосредственно с помощью меню УЦ.

Программа Service дает возможность:

создавать и изменять структуры БД;

открывать, закрывать и переиндексировать файлы БД;

редактировать введенные данные и добавлять записи;

просматривать файлы БД и удалять записи;

осуществлять поиск, отбор и сортировку данных;

выполнять вывод данных на экран, принтер или в файл;

устанавливать параметры и состояния dBASE IV;

осуществлять экспорт-импорт файлов;

вычислять размер свободной оперативной памяти;

выполнять некоторые команды DOS;

осуществлять выход из программы в dBASE IV или DOS.

После входа в программу высвечивается основное горизонтальное меню с позициями "**Файл БД**", "**Данные**", "**Поиск**", "**Вывод**", "**Сервис**", "**Выход**", к которым прикрепляются многоуровневые вертикальные меню.

Ниже на рис. 5.1 показана общая блок-схема программы Service, а на рис. 5.2...5.4 приведены блок-схемы ее ветвей, реализующие функции позиций вертикальных меню, прикрепленных к позициям основного горизонтального меню.

Функции меню реализуются посредством соответствующей процедуры, имя которой помещено в блок-схеме над меню. Процедуры меню обеспечивают переход на следующий уровень или вызов исполняемой процедуры.
При переходе на меню ниже следующего уровня вызывается процедура, связанная с этим меню.

При переходе между позициями горизонтального меню автоматически высвечиваются прикрепленные к ним вертикальные меню. Активизация позиций вертикальных меню нижних уровней выполняется нажатием клавиши Enter.

Программа Service состоит из главной процедуры и вызываемых процедур, выполняющих операции, связанные с выбранной позицией меню. В начале главной процедуры выполняются команды по установке параметров и состояний dBASE IV.

Этими командами назначается частота и длительность звукового сигнала, подаваемого при заполнении последнего символа в поле записи файла БД; определяется тип рамки вокруг вертикальных меню; отменяется режим копирования содержимого предыдущей записи при переходе к новой записи файла БД; закрывается текущий каталог; восстанавливается расцветка экрана, задаваемая в dBASE IV по умолчанию; отменяется вывод часов на экране; выключается функция аварийного прерывания выполнения команд клавишей ESC и отменяется вывод на печать или экран выполняемых команд и их результатов; подавляется отображение на экране информационной строки текущего состояния системы dBASE IV.

После установок системных параметров командой PUBLIC определяются глобальные переменные, которые будут использоваться повсюду в программе. Некоторым из них присваиваются начальные значения. Например, символьной переменной mess присваивается справочный текст по использованию меню, который после выбора соответствующей позиции меню высвечивается в строке сообщений экрана.

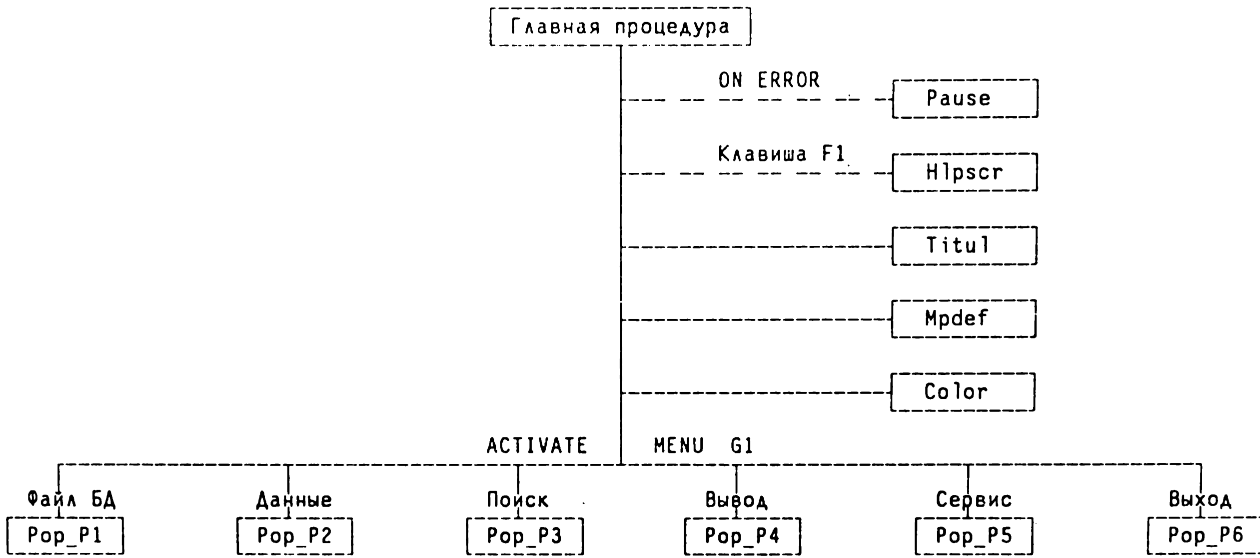


Рис. 5.1. Общая блок-схема программы Service

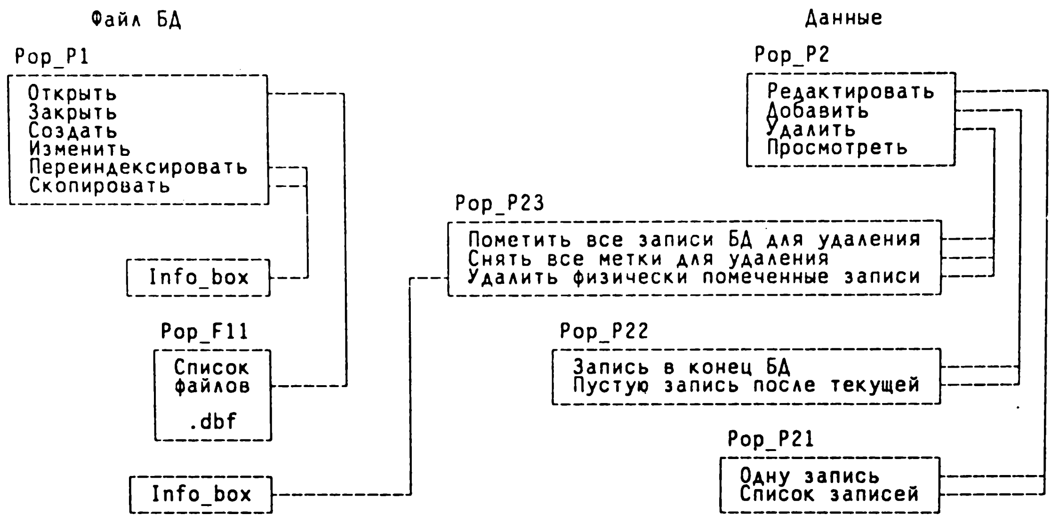


Рис. 5.2. Блок-схемы ветвей программы Service "Файл БД" и "Данные"

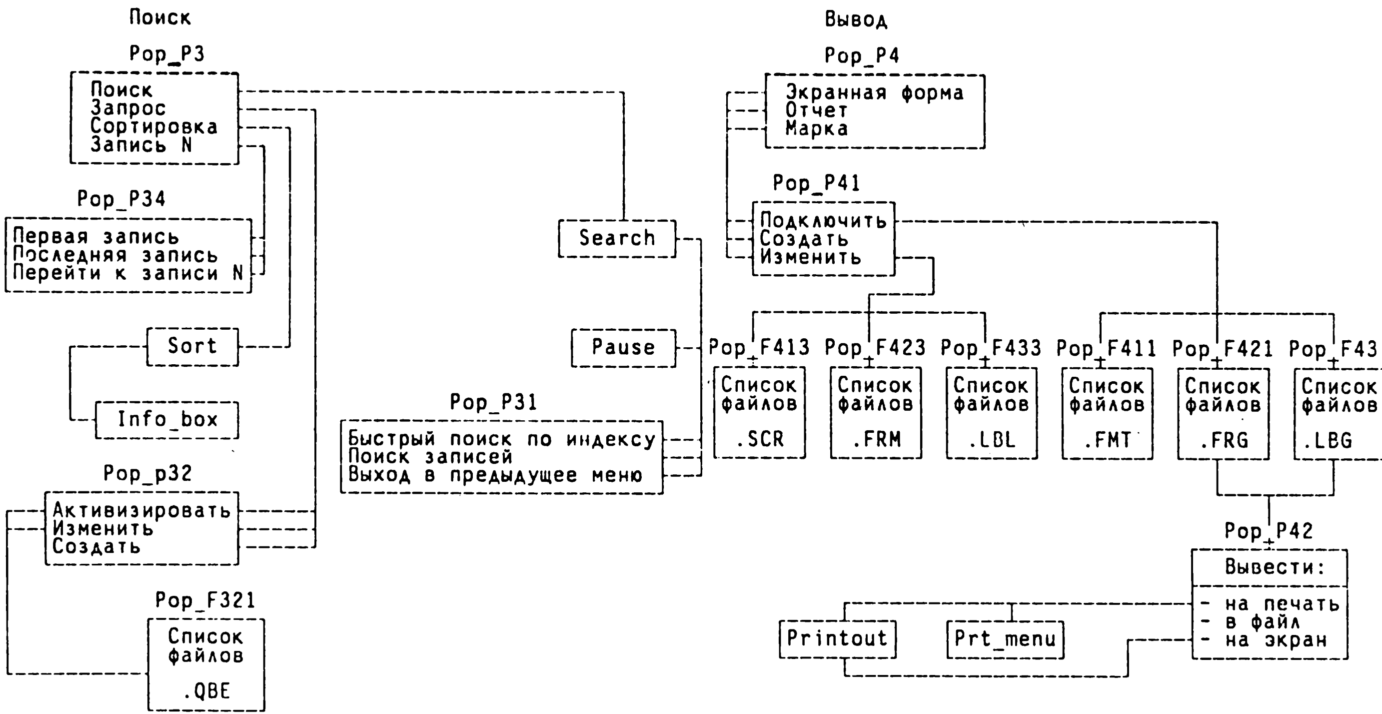


Рис. 5.3. Блок-схемы ветвей программы Service "Поиск" и "Вывод"

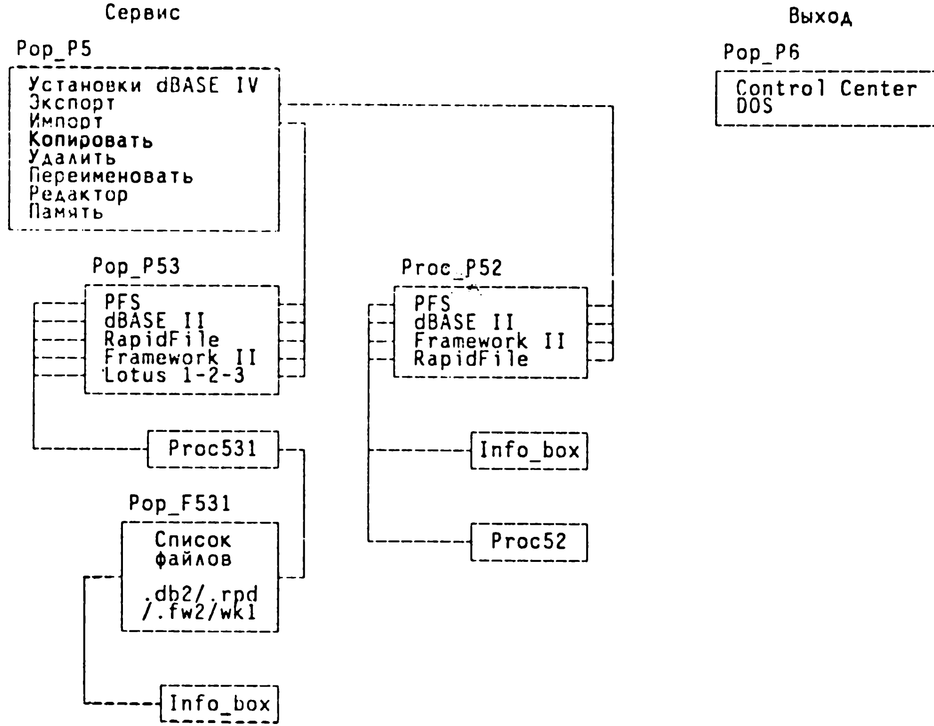


Рис. 5.4. Блок-схемы ветвей программы Service "Сервис" и "Выход"

```
* Service.prg
SET BELL TO 60,10
SET BORDER TO SINGLE
SET CARRY OFF
SET CATALOG TO
SET COLOR TO
SET CLOCK OFF
SET ESCAPE OFF
SET ECHO OFF
SET PRINT OFF
SET TALK OFF
SET SCOREBOARD OFF
SET STATUS OFF
PUBLIC ln_type,ln_rkey,ln_rec,ln_num,ld_date,lc_option,;
      fr,gn_error,gn_ikey,gn_send,gc_key,listval,mess
fr=space(8)
gn_error=0
gn_ikey=0
gn_send=0
gc_key='H'
listval='NO_FIELD'
mess='Переход между позициями: '+CHR(27)+CHR(26)+' ';
+CHR(25)+CHR(24)+'  Выбор позиции: ';
+CHR(17)+CHR(196)+CHR(217)+'  Помощь: F1'
```

Команда DEFINE WINDOW определяет окно с именем Pause, которое в дальнейшем будет использоваться для вывода сообщений об ошибках или различных комментариев. Обработка ошибочной ситуации выполняется командой ON ERROR. Эта команда в случае такой ситуации вызывает процедуру Pause, передавая ей через параметры имя процедуры и номер строки, в которой произошла ошибка. Команда ON KEY LABEL назначает клавише F1 вызов с помощью процедуры Hlpscr на экран справочной информации. Затем последовательно запускаются процедуры: Titul (вывод заставки программы Service); Mpdef (определение окон, горизонтального и вертикальных меню) и Color (расцветка экрана).

```
DEFINE WINDOW Pause FROM 15,5 TO 19,75 NONE COLOR W+/R
ON ERROR DO Pause WITH 'Процедура: '+Program()+;
' Строка : '+LTRIM(STR(LINE()))
ON KEY LABEL F1 DO Hlpscr
DO Titul
DO Mpdef
```

В составе главной процедуры основной командой является команда цикла DO WHILE...ENDDO, в теле которой находится команда ACTIVATE MENU G1, активизирующая основное горизонтального меню G1 с прикрепленными к его позициям вертикальными меню. Условием входа в цикл служит значение переменной gc_key='H'. После выбора позиции "Dbase IV" вертикального меню P6, прикрепленного к позиции "Выход", в процедуре Pop_P6, которая вызывается при активизации меню P6, командой DEACTIVATE MENU деактивизируется меню G1 и активизируется окно Exit_App (в нем высвечивается предложение ввести букву "Д" для выхода из программы). При вводе вуквы "Д" осуществляется выход из цикла DO WHILE...ENDDO и управление передается на завершающую часть главной процедуры. В противном случае снова активизируется горизонтальное меню G1. При выборе позиции "DOS" меню "Выход" переход в DOS осуществляется без высвечивания сообщения.

В завершающей части главной процедуры закрываются все файлы и процедуры, оперативная память очищается от переменных, меню и окон,

клавишам F1 и Esc возвращаются их первоначальные функции и команда RETURN осуществляет выход из программы.

```
DO WHILE gc_key ='H'
  DO COLOR
  SET STATUS ON
  ACTIVATE MENU G1
  DEFINE WINDOW Exit_App FROM 9,17 TO 12,62 NONE COLOR W+/R
  SET MESSAGE TO;
  'Для выбора нужного ответа (Да/Нет) нажмите Spacebar '
  ACTIVATE WINDOW Exit_App
  @ 1,1 SAY "Вы хотите завершить работу с системой ? ";
  GET gc_key PICT "@M Д,Н"
  READ
  CLEAR GETS
  SET MESSAGE TO
  DEACTIVATE WINDOW Exit_App
ENDDO

ON KEY LABEL F1
CLEAR ALL
CLEAR WINDOWS
CLEAR POPUPS
CLOSE ALL
CLOSE PROCEDURE
SET ESCAPE ON
SET MESSAGE TO
CLEAR
RETURN && Выход из программы Service

```

Вторая часть программы Service содержит вызываемые процедуры, которые разделяются на четыре группы. К первой группе относятся процедуры, которые вызываются непосредственно из главной процедуры и упомянуты выше. Вторая группа включает процедуры, вызываемые при активизации вертикальных меню различных уровней. Эти процедуры имеют такие же имена, как и меню, но с приставкой впереди Pop_.

В состав третьей группы входят процедуры, вызываемые при выполнении определенных позиций меню: Search, Sort, Prt_menu, Printout. Четвертая группа состоит из процедур, имеющих общее назначение, которые вызываются для реализации определенных функций: Info_box, Pause, Hlpscr и Bell. Схема вызова процедур приведена на рис. 5.1...5.4.

Вызываемые процедуры могут располагаться во второй части программы в любой последовательности. Однако для лучшего понимания работы программы Service целесообразно в начале поместить процедуру Mpdef.

Процедура Mpdef определяет сначала окна, затем горизонтальное и вертикальные меню, используемые в программе. Окно Savescr предназначено для вывода различной информации на полном (по ширине) экране. В окне Helpscr выводится справочная информации по использованию меню. Окно Browscr используется для вывода записей файла БД в форме Browse, а окно Pause - для вывода сообщения об ошибке. В окне Mprint выводятся параметры, устанавливаемые перед печатью отчета или марки. В окне Tfile задается имя текстового файла при выводе в него отчета или марки.

В процедуре Mpdef после определения порции окон и меню командой @...SAY в строке сообщений экрана высвечивается слово "Загрузка..." и движущиеся символы ░, отражающие течение процесса определения меню и окон. По завершению этого процесса появляется надпись "100 %".

```
PROCEDURE Mpdef
DEFINE WINDOW FullScr FROM 0,0 TO 24,79 NONE
DEFINE WINDOW Savescr FROM 0,0 TO 21,79 NONE
DEFINE WINDOW Helpscr FROM 3,0 TO 19,79 NONE COLOR BG+/B
DEFINE WINDOW Browscr FROM 1,0 TO 21,79 NONE
DEFINE WINDOW Mprint FROM 4,5 TO 20,67 NONE COLOR BG+/B
DEFINE WINDOW Tfile FROM 12,3 TO 15,37 NONE COLOR GR+/G+
@ 23,20 SAY "Загрузка..."
SET COLOR TO R+

```

Затем определяется горизонтальное меню G1, его позиции и назначается активизация вертикальных меню при выборе этих позиций. Например, при выборе позиций "Файл БД" основного (горизонтального) меню активизируется вертикальное меню P1.

```
DEFINE MENU G1
DEFINE PAD Pad_1 OF G1 PROMPT "Файл БД" AT 1,5
DEFINE PAD Pad_2 OF G1 PROMPT "Данные " AT 1,18
DEFINE PAD Pad_3 OF G1 PROMPT " Поиск " AT 1,30
DEFINE PAD Pad_4 OF G1 PROMPT " Вывод " AT 1,43
DEFINE PAD Pad_5 OF G1 PROMPT "Сервис " AT 1,57
DEFINE PAD Pad_6 OF G1 PROMPT " Выход " AT 1,68
ON PAD Pad_1 OF G1 ACTIVATE POPUP P1
ON PAD Pad_2 OF G1 ACTIVATE POPUP P2
ON PAD Pad_3 OF G1 ACTIVATE POPUP P3
ON PAD Pad_4 OF G1 ACTIVATE POPUP P4
ON PAD Pad_5 OF G1 ACTIVATE POPUP P5
ON PAD Pad_6 OF G1 ACTIVATE POPUP P6

```

В вертикальном меню P1 определяется восемь позиций, из которых активными являются только шесть. Третья и седьмая позиции неактивны (при движении курсора эти позиции пропускаются), на что указывает опция SKIP в команде DEFINE BAR. При активизации меню в информационной строке экрана высвечивается сообщение, записанное в переменной mess. Реализация функций меню P1 выполняется в процедуре Pop_P1.

```
DEFINE POPUP P1 FROM 3,2 TO 12,23 MESSAGE mess
DEFINE BAR 1 OF P1 PROMPT " Открыть            "
DEFINE BAR 2 OF P1 PROMPT " Закрыть            "
DEFINE BAR 3 OF P1 PROMPT "————————————————————" SKIP
DEFINE BAR 4 OF P1 PROMPT " Создать            "
DEFINE BAR 5 OF P1 PROMPT " Изменить           "
DEFINE BAR 6 OF P1 PROMPT " Переиндексировать  "
DEFINE BAR 7 OF P1 PROMPT "————————————————————" SKIP
DEFINE BAR 8 OF P1 PROMPT " Скопировать        "
ON SELECTION POPUP P1 DO Pop_P1

```

Затем определяется вертикальное меню F11, в котором высвечивается список файлов заданного типа, определяемого опцией LIKE команды DEFINE POPUP. При активизации меню F11 осуществляется вызов процедуры Pop_F11. Далее определяются вертикальные меню P2, P21, P22, P23, P3 и др.

```
DEFINE POPUP F11 FROM 5,18 TO 20,33 PROMPT FILES LIKE *.dbf
ON SELECTION POPUP F11 DO Pop_F11
?? CHR(176)

DEFINE POPUP P2 FROM 3,18 TO 11,38 MESSAGE mess
DEFINE BAR 1 OF P2 PROMPT "  Редактировать  "
```

```
DEFINE BAR 2 OF P2 PROMPT "────────────────────" SKIP
DEFINE BAR 3 OF P2 PROMPT "  Добавить        "
DEFINE BAR 4 OF P2 PROMPT "────────────────────" SKIP
DEFINE BAR 5 OF P2 PROMPT "  Удалить         "
DEFINE BAR 6 OF P2 PROMPT "────────────────────" SKIP
DEFINE BAR 7 OF P2 PROMPT "  Просмотреть     "
ON SELECTION POPUP P2 DO Pop_P2
?? CHR(176)

DEFINE POPUP P21 FROM 6,31 MESSAGE mess
DEFINE BAR 1 OF P21 PROMPT "  Одну запись  (Edit)       "
DEFINE BAR 2 OF P21 PROMPT "─────────────────────────────" SKIP
DEFINE BAR 3 OF P21 PROMPT "  Список записей  (Browse) "
ON SELECTION POPUP P21 DO Pop_P21
?? CHR(176)

DEFINE POPUP P22 FROM 6,31 TO 10,63  MESSAGE mess
DEFINE BAR 1 OF P22 PROMPT " Запись в конец БД          "
DEFINE BAR 2 OF P22 PROMPT "─────────────────────────────" SKIP
DEFINE BAR 3 OF P22 PROMPT " Пустую запись после текущей"
ON SELECTION POPUP P22 DO Pop_P22
?? CHR(176)

DEFINE POPUP P23 FROM 7,31 TO 13,71  MESSAGE mess
DEFINE BAR 1 OF P23 PROMPT "Пометить все удаляемые записи"
DEFINE BAR 2 OF P23 PROMPT "──────────────────────────────" SKIP
DEFINE BAR 3 OF P23 PROMPT " Снять все метки для удаления"
DEFINE BAR 4 OF P23 PROMPT "──────────────────────────────" SKIP
DEFINE BAR 5 OF P23 PROMPT " Удалить помеченные записи   "
ON SELECTION POPUP P23 DO Pop_P23
?? CHR(176)

DEFINE POPUP P3 FROM 3,30 TO 11,47  MESSAGE mess
DEFINE BAR 1 OF P3 PROMPT "  Поиск         "
DEFINE BAR 2 OF P3 PROMPT "────────────────" SKIP
DEFINE BAR 3 OF P3 PROMPT "  Запрос        "
DEFINE BAR 4 OF P3 PROMPT "────────────────" SKIP
DEFINE BAR 5 OF P3 PROMPT "  Сортировка  "
DEFINE BAR 6 OF P3 PROMPT "────────────────" SKIP
DEFINE BAR 7 OF P3 PROMPT "  Запись N      "
ON SELECTION POPUP P3 DO Pop_P3
?? CHR(176)

DEFINE POPUP P32 FROM 6,43  MESSAGE mess
DEFINE BAR 1 OF P32 PROMPT "  Активизировать "
DEFINE BAR 2 OF P32 PROMPT "──────────────────" SKIP
DEFINE BAR 3 OF P32 PROMPT "  Изменить       "
DEFINE BAR 4 OF P32 PROMPT "──────────────────" SKIP
DEFINE BAR 5 OF P32 PROMPT "  Создать        "
ON SELECTION POPUP P32 DO Pop_P32
?? CHR(176)

DEFINE POPUP P34 FROM 8,44 TO 14,71  MESSAGE mess
DEFINE BAR 1 OF P34 PROMPT "  Первая запись         "
DEFINE BAR 2 OF P34 PROMPT "────────────────────────" SKIP
DEFINE BAR 3 OF P34 PROMPT "  Последняя запись      "
DEFINE BAR 4 OF P34 PROMPT "────────────────────────" SKIP
DEFINE BAR 5 OF P34 PROMPT "  Перейти к записи N    ";
MESSAGE 'Введите номер записи'
ON SELECTION POPUP P34 DO Pop_P34
?? CHR(176)

```

```
DEFINE POPUP F321 FROM 9,55 TO 19,70 PROMPT FILES LIKE *.QBE
ON SELECTION POPUP F321 DO Pop_F321 WITH AA
?? CHR(176)

DEFINE POPUP P4 FROM 3,43 TO 9,63  MESSAGE mess
DEFINE BAR 1 OF P4 PROMPT " Экранная форма  "
DEFINE BAR 2 OF P4 PROMPT "────────────────────" SKIP
DEFINE BAR 3 OF P4 PROMPT " Отчет         "
DEFINE BAR 4 OF P4 PROMPT "────────────────────" SKIP
DEFINE BAR 5 OF P4 PROMPT " Марка         "
ON SELECTION POPUP P4 DO Pop_P4
?? CHR(176)

DEFINE POPUP P41 FROM 5,55 TO 11,71  MESSAGE mess
DEFINE BAR 1 OF P41 PROMPT " Подключить  "
DEFINE BAR 2 OF P41 PROMPT "──────────────" SKIP
DEFINE BAR 3 OF P41 PROMPT " Создать     "
DEFINE BAR 4 OF P41 PROMPT "──────────────" SKIP
DEFINE BAR 5 OF P41 PROMPT " Изменить    "
ON SELECTION POPUP P41 DO Pop_P41 WITH BB
?? CHR(176)

DEFINE POPUP P42 FROM 11,43  MESSAGE mess
DEFINE BAR 1 OF P42 PROMPT "   В ы в е с т и :  " SKIP
DEFINE BAR 2 OF P42 PROMPT "──────────────────────" SKIP
DEFINE BAR 3 OF P42 PROMPT " - на  печать       "
DEFINE BAR 4 OF P42 PROMPT "──────────────────────" SKIP
DEFINE BAR 5 OF P42 PROMPT " - в текстовый файл "
DEFINE BAR 6 OF P42 PROMPT "──────────────────────" SKIP
DEFINE BAR 7 OF P42 PROMPT " - на экран         "
ON SELECTION POPUP P42 DO Pop_P42
?? CHR(176)

DEFINE POPUP F411 FROM 9,63 TO 20,77 PROMPT FILES LIKE *.fmt
ON SELECTION POPUP F411 DO Pop_F411
?? CHR(176)

DEFINE POPUP F413 FROM 9,63 TO 20,77 PROMPT FILES LIKE *.scr
ON SELECTION POPUP F413 DO Pop_F413
?? CHR(176)

DEFINE POPUP F421 FROM 9,63 TO 20,77 PROMPT FILES LIKE *.frg
ON SELECTION POPUP F421 DO Pop_F421
?? CHR(176)

DEFINE POPUP F423 FROM 9,63 TO 20,77 PROMPT FILES LIKE *.frm
ON SELECTION POPUP F423 DO Pop_F423
?? CHR(176)

DEFINE POPUP F431 FROM 9,63 TO 20,77 PROMPT FILES LIKE *.lbg
ON SELECTION POPUP F431 DO Pop_F431
?? CHR(176)

DEFINE POPUP F433 FROM 9,63 TO 20,77 PROMPT FILES LIKE *.lbl
ON SELECTION POPUP F433 DO Pop_F433
?? CHR(176)

DEFINE POPUP P5 FROM 3,48  MESSAGE mess
DEFINE BAR  1 OF P5 PROMPT " Установки dBASE IV  "
DEFINE BAR  2 OF P5 PROMPT "──────────────────────" SKIP
DEFINE BAR  3 OF P5 PROMPT " Экспорт             "
DEFINE BAR  4 OF P5 PROMPT " Импорт              "
```

```
DEFINE BAR  5 OF P5 PROMPT "──────────────────────────" SKIP
DEFINE BAR  6 OF P5 PROMPT " Копировать           "
DEFINE BAR  7 OF P5 PROMPT "──────────────────────────" SKIP
DEFINE BAR  8 OF P5 PROMPT " Удалить              "
DEFINE BAR  9 OF P5 PROMPT "──────────────────────────" SKIP
DEFINE BAR 10 OF P5 PROMPT " Переименовать        "
DEFINE BAR 11 OF P5 PROMPT "──────────────────────────" SKIP
DEFINE BAR 12 OF P5 PROMPT " Редактор текста      "
DEFINE BAR 13 OF P5 PROMPT "──────────────────────────" SKIP
DEFINE BAR 14 OF P5 PROMPT " Память               "
ON SELECTION POPUP P5 DO Pop_P5
?? CHR(176)

DEFINE POPUP P52 FROM 7,60 TO 15,77 MESSAGE mess
DEFINE BAR 1 OF P52 PROMPT "√PFS          "
DEFINE BAR 2 OF P52 PROMPT "───────────────" SKIP
DEFINE BAR 3 OF P52 PROMPT " dBASE II     "
DEFINE BAR 4 OF P52 PROMPT "───────────────" SKIP
DEFINE BAR 5 OF P52 PROMPT " Framework II "
DEFINE BAR 6 OF P52 PROMPT "───────────────" SKIP
DEFINE BAR 7 OF P52 PROMPT " RapidFile    "
ON SELECTION POPUP P52 DO Pop_P52
?? " 100 %"

DEFINE POPUP P53 FROM 7,60 TO 17,77  MESSAGE mess
DEFINE BAR 1 OF P53 PROMPT " PFS"
DEFINE BAR 2 OF P53 PROMPT "───────────────" SKIP
DEFINE BAR 3 OF P53 PROMPT " dBASE II"
DEFINE BAR 4 OF P53 PROMPT "───────────────" SKIP
DEFINE BAR 5 OF P53 PROMPT " RapidFile"
DEFINE BAR 6 OF P53 PROMPT "───────────────" SKIP
DEFINE BAR 7 OF P53 PROMPT " Framework II"
DEFINE BAR 8 OF P53 PROMPT "───────────────" SKIP
DEFINE BAR 9 OF P53 PROMPT " WK1"
ON SELECTION POPUP P53 DO Pop_P53

DEFINE POPUP P6 FROM 3,60 TO 7,79  MESSAGE mess
DEFINE BAR 1 OF P6 PROMPT " Control Center "
DEFINE BAR 2 OF P6 PROMPT "─────────────────" SKIP
DEFINE BAR 3 OF P6 PROMPT " DOS            "
ON SELECTION POPUP P6 DO Pop_P6
RETURN
```

За процедурой Mpdef помещаются процедуры, реализующие функции прикладной программы Service, которые вызываются при активизации вертикальных меню. Процедура Pop_P1 вызывается при активизации вертикального меню P1 и реализует функции открытия, закрытия, создания и переиндексации файла БД, изменения и копирования его структуры. Основой процедуры служит команда DO CASE... ENDCASE, позволяющая разветвлять управление в зависимости от выбранной позиции меню. Например, после выбора позиции "Открыть" управление передается на команду ACTIVATE POPUP F11 после опции CASE BAR() = 1, в результате чего активизируется вертикальное меню F11, которое определено в процедуре Mpdef. С помощью этого меню пользователь выбирает файл, который нужно открыть.

```
PROCEDURE Pop_P1
DO CASE
  CASE BAR() = 1 && Открыть
    ACTIVATE POPUP F11
  CASE BAR() = 2 && Закрыть
```

```
    ACTIVATE WINDOW Savescr
    USE
    DEACTIVATE WINDOW Savescr
  CASE BAR() = 4 && Создать
    ACTIVATE WINDOW Savescr
    bd =SPACE(8)
    @ 5,15 to 7,60
    @ 6,18 SAY "Введите имя новой базы данных  " GET bd
    READ
    CREATE &bd
    CLEAR GETS
    DEACTIVATE WINDOW Savescr
  CASE BAR() = 5 && Изменить
    ACTIVATE WINDOW Savescr
    MODIFY STRUCTURE
    DEACTIVATE WINDOW Savescr
  CASE BAR() = 6 && Переиндексировать
    ACTIVATE WINDOW Savescr
    lc_say =' Переиндексация базы данных...'
    DO info_box WITH lc_say
    SET TALK ON
    REINDEX
    SET TALK OFF
    DEACTIVATE WINDOW Savescr
  CASE BAR() = 8 && Копировать
    ACTIVATE WINDOW Savescr
    bd =SPACE(8)
    @ 5,15 to 7,60
    @ 6,17 SAY "Введите имя базы данных  " GET bd
    READ
    CLEAR GETS
    COPY STRUCTURE TO &bd
    DEACTIVATE WINDOW Savescr
ENDCASE
SET MESSAGE TO '
DEACTIVATE POPUP
RETURN
```

Процедура Pop _ F11 вызывается после выбора позиции "Открыть" вертикального меню P1 при активизации меню F11 и реализует функцию открытия существующего файла. Функция PROMPT() возвращает имя файла БД с маршрутом по директориям, выбранным из списка в окне вертикального меню F11, которое присваивается переменной listval, и затем команда USE &listval открывает выбранный файл БД. Знак & функции макроподстановки означает, что в качестве имени открываемого файла используется не имя переменной, а ее содержимое. После открытия файла БД команда DEACTIVATE POPUP деактивизирует вертикальное меню F11, а команда RETURN выполняет возврат к процедуре Pop _ P1.

```
PROCEDURE Pop_F11
listval =PROMPT()
USE &listval
SET MESSAGE TO
DEACTIVATE POPUP
RETURN
```

Процедура Pop _ P2 по своей структуре подобна процедуре Pop _ P1 и выполняет функции вертикального меню P2 по редактированию, добавлению, удалению и просмотру записей открытого файла БД.

```
PROCEDURE Pop_P2
DO CASE
  CASE BAR() = 1 && Редактировать
    ACTIVATE POPUP P21
  CASE BAR() = 3 && Добавить
    ACTIVATE POPUP P22
  CASE BAR() = 5 && Удалить
    ACTIVATE POPUP P23
  CASE BAR() = 7 && Просмотреть
    ACTIVATE WINDOW Savescr
    CLEAR
    DISPLAY ALL
    SET MESSAGE TO;
    "Для продолжения работы нажмите любую клавишу..."
    SET CONSOLE OFF
    WAIT
    SET CONSOLE ON
    DEACTIVATE WINDOW Savescr
ENDCASE
SET MESSAGE TO
DEACTIVATE POPUP
RETURN

```

При выборе позиции "Редактировать" вертикального меню P2 активизируется вертикальное меню P21 с двумя активными позициями. Реализация функций меню P21 по редактированию записей файла БД в стандартных экранных формах Edit или Browse выполняется в процедуре Pop_P21.

```
PROCEDURE Pop_P21
DO CASE
  CASE BAR() = 1 && Edit-форма
    IF .not.FR =SPACE(8)
      SET FORMAT TO &FR
    ENDIF
    ACTIVATE WINDOW Browscr
    EDIT
    SET FORMAT TO
    DEACTIVATE WINDOW Browscr
  CASE BAR() = 3 && Browse-таблица
    ACTIVATE WINDOW Browscr
    BROWSE
    DEACTIVATE WINDOW Browscr
ENDCASE
SET MESSAGE TO
DEACTIVATE POPUP
RETURN

```

Процедура Pop_P22 вызывается при выборе позиции "Добавить" вертикального меню P2 и реализует функции по добавлению записей в конец файла БД (командой APPEND) или же после текущей записи (командами INSERT BLANK и EDIT).

```
PROCEDURE Pop_P22
DO CASE
  CASE BAR() = 1 && Добавить запись в конец БД
    ACTIVATE WINDOW Browscr
    APPEND
    DEACTIVATE WINDOW Browscr
  CASE BAR() = 3 && Добавить пустую запись после текущей
```

```
    ACTIVATE WINDOW Savescr
    INSERT BLANK
    EDIT
    DEACTIVATE WINDOW Savescr
ENDCASE
SET MESSAGE TO
DEACTIVATE POPUP
RETURN
```

Удаление записей файлов БД в dBASE IV выполняется в два этапа: пометка записей, предназначенных для удаления, и физическое удаление помеченных записей. Процедура Pop _ P23 выполняет пометку записей для последующего удаления, уничтожение этих пометок и физическое удаление помеченных записей файла БД. Вызывается при активизации вертикального меню P23.

```
PROCEDURE Pop_P23
DO CASE
  CASE BAR() = 1 && Пометка записей для удаления
    ACTIVATE WINDOW Savescr
    lc_say ='Пометка записей для удаления...'
    DO info_box WITH lc_say
    SET TALK ON
    DELETE ALL
    SET TALK OFF
    DEACTIVATE WINDOW Savescr
  CASE BAR() = 3 && Снять все метки для удаления записей
    ACTIVATE WINDOW Savescr
    lc_say ='Восстановление записей...'
    DO info_box WITH lc_say
    SET TALK ON
    RECALL ALL
    SET TALK OFF
    DEACTIVATE WINDOW Savescr
  CASE BAR() = 5 && Удалить физически помеченные записи
    ACTIVATE WINDOW Savescr
    lc_say ='Физическое удаление записей...'
    DO info_box WITH lc_say
    SET TALK ON
    PACK
    SET TALK OFF
    GO TOP
    DEACTIVATE WINDOW Savescr
ENDCASE
SET MESSAGE TO
DEACTIVATE POPUP
RETURN
```

Поиск информации в файле БД по заданному условию, отбор данных, сортировка записей, а также переход к записи по номеру выполняются с помощью вертикального меню P3, прикрепленного к позиции "Поиск" горизонтального меню G1. Функции меню P3 реализует процедура Pop _ P3. При выборе позиции "Поиск" вертикального меню осуществляется вызов процедуры поиска Search. Выбор позиции "Запрос" активизирует вертикальное меню P32, функции которого реализуются процедурой Pop _ P32.

Для сортировки записей после выбора позиции "Сортировка" выполняется процедура Sort. Выбор позиции "Запись N" меню активизирует вертикальное меню P34, которое позволяет перейти к первой и последней записи, а также к записи с заданным номером.

```
PROCEDURE Pop_P3
DO CASE
  CASE BAR() = 1 && Поиск
    DO SEARCH
  CASE BAR() = 3 && Запрос
    ACTIVATE POPUP P32
  CASE BAR() = 5 && Сортировка
    DO SORT
  CASE BAR() = 7 && Запись N
    ACTIVATE POPUP P34
ENDCASE
SET MESSAGE TO
DEACTIVATE POPUP
RETURN
```

Процедура Search выполняет два типа поиска: быстрый по индексному полю (выражению) и поиск записей по логическому выражению, составленному в соответствии с правилами отбора записей в языке dBASE IV.

Быстрый поиск осуществляется по активному в данный момент индексному полю (выражению) посредством ввода поискового значения, а поиск по условию выполняется с использованием сформированного пользователем выражения, содержащегося в переменных lc_scp (диапазон поиска), lc_ln2 (критерий поиска), ln_ln3 (условие окончания поиска), и записанного в переменную lc_temp.

```
PROCEDURE SEARCH
IF LEN(DBF()) = 0
  DO Pause WITH "           НЕТ открытой базы  данных"
  RETURN
ENDIF
SET SPACE ON
SET DELIMITERS OFF
ln_type=0          && Выбор подуровня
ln_rkey=READKEY()  && Тест для ESC или Return
ln_rec=RECNO()     && Номер записи
ln_num=0           && Для ввода числа
ld_date=DATE()     && Для ввода даты
lc_option='0'      && Главная опция (Seek, Goto and Locate)

STORE SPACE(10) TO lc_scp
STORE SPACE(40) TO lc_ln1, lc_ln2, lc_ln3
lc_temp=""
ACTIVATE WINDOW Savescr
@ 0,1 SAY "Подключен индекс по полю : "+IIF(""=ORDER(),;
"БД используется без индексного файла",ORDER());
COLOR GR+/N
lc_temp=REPLICATE(CHR(196),19)
@ 2,0 SAY CHR(218)+lc_temp+CHR(194)+lc_temp+CHR(194)+;
lc_temp+CHR(194)+lc_temp
ln_num=240

DO WHILE ln_num < 560
  lc_temp=FIELD( (ln_num-240)/20 +1)
  @ (ln_num/80),MOD(ln_num,80) SAY CHR(179)+;
  lc_temp+SPACE(11-LEN(lc_temp))+;
  SUBSTR("= Симв. = Дата  = Логич.= Число = Float = Memo  ",;
  AT(TYPE(lc_temp),"CDLNFMU")*8-7,8)
  ln_num=ln_num+20
ENDDO
```

```
ln_num=1
DEFINE POPUP P31 FROM 8,25
DEFINE BAR 1 OF P31 PROMPT " Быстрый поиск по индексу ";
MESSAGE " Поиск по индексному полю " SKIP FOR ""=ORDER()
DEFINE BAR 2 OF P31 PROMPT "————————————————" SKIP
DEFINE BAR 3 OF P31 PROMPT " Поиск записей          ";
MESSAGE "Поиск записей по заданному условию "
DEFINE BAR 4 OF P31 PROMPT "————————————————" SKIP
DEFINE BAR 5 OF P31 PROMPT " Выход в предыдущее меню  ";
MESSAGE "Для выхода нажмите клавишу Enter "
ON SELECTION POPUP P31 DO Pop_P31
SET CONFIRM ON

DO WHILE lc_option='0'
  ACTIVATE POPUP P31
  lc_option = ltrim(str(gn_send))  && Для верт. меню

  IF LASTKEY() = 27 .OR. lc_option="5"
    GOTO ln_rec
    EXIT
  ENDIF

  DO CASE
    CASE lc_option='1' && Быстрый поиск
      IF LEN(NDX(1))=0 .AND. LEN(MDX(1))=0
        DO Pause WITH "Нет открытых индексных файлов"
        LOOP
      ENDIF

      ln_type=" "
      lc_ln1=SPACE(45)
      DEFINE WINDOW Posit2 FROM 8,13 TO 15,62;
      NONE COLOR W+/B+
      ACTIVATE WINDOW Posit2
      @ 1,2 SAY "Нажатием клавиши  SpaceBar  выберите тип поля"
      @ 2,5 SAY "(1 - символ, 2 - число  и  3 = дата)";
      GET ln_type PICTURE "@M 1,2,3" COLOR W/B+
      READ

      IF .NOT. (READKEY() = 12 .OR. READKEY() = 268)
        SET CONFIRM ON
        @ 4,14 SAY "Введите критерий поиска " COLOR BG+*/B+

        IF ln_type='3'
          @ 6,2 GET ld_date PICT "@D"
        ELSE

          IF ln_type='2'
            @ 6,2 GET ln_num PICT "##########"
          ELSE
            @ 6,2 GET lc_ln1
          ENDIF

        ENDIF
        READ
        SET CONFIRM OFF

        IF .NOT. (READKEY() = 12 .OR. READKEY() = 268)
          lc_temp=IIF(ln_type='1',"TRIM(lc_ln1)",;
          IIF(ln_type='2',"ln_num","ld_date"))
          SEEK &lc_temp.
        ENDIF
```

```
  ENDIF
  RELEASE WINDOWS Posit2
  CASE lc_option='3' && Поиск
    DEFINE WINDOW Posit2 FROM 8,8 TO 14,77;
    NONE COLOR W+/BR+
    ACTIVATE WINDOW Posit2
    @ 1,22 SAY " (ALL, NEXT <n>, and REST)"
    @ 1,01 SAY "Диапазон:" GET lc_scp
    @ 3,01 SAY "Критерий поиска: " GET lc_ln2
    @ 5,01 SAY "Условие окончания поиска:" GET lc_ln3
    READ

    IF .NOT. (READKEY() = 12 .OR. READKEY() = 268)
      lc_temp=TRIM(lc_scp)
      lc_temp=lc_temp + IIF(LEN(TRIM(lc_ln2)) > 0,";
      FOR "+TRIM(lc_ln2),"")
      lc_temp=lc_temp + IIF(LEN(TRIM(lc_ln3)) > 0,";
      WHILE "+TRIM(lc_ln3),"")

      IF LEN(lc_temp) > 0
        LOCATE &lc_temp.
        EDIT
      ELSE
        DO Pause WITH "ВЫ не заполнили ни одной позиции..."
      ENDIF

    ENDIF
    RELEASE WINDOW Posit2
 ENDCASE
 IF EOF()
    DO Pause WITH "Запись не найдена ..."
    GOTO ln_rec
 ENDIF

 IF READKEY()=12 .OR. READKEY()= 268 .OR. LASTKEY()=27
    lc_option='0'
 ENDIF

ENDDO
DEACTIVATE WINDOW Savescr
SET CONFIRM OFF
CLEAR GETS
RETURN
```

Процедура Pop_P31 определяет номер выбранной позиции вертикального меню с именем P31 и присваивает этот номер переменной gn_send.

```
PROCEDURE Pop_p31
gn_send = BAR()
DEACTIVATE POPUP
RETURN
```

Выбор позиции "Запрос" вертикального меню P3 обеспечивает активизацию вертикального меню P32, функции которого реализованы в процедуре Pop_P32. Меню P32 имеет пять позиций, три из которых (первая "Активизировать", третья "Изменить" и пятая "Создать") являются активными. При выборе первой или третьей позиции меню вызывается одна и та же процедура Pop_F321. Выбрав пятую позицию, пользователь получает возможность создать новый файл запроса.

Экран создания запроса на выборку данных вызывается командой CREATE QUERY.

```
PROCEDURE Pop_P32
DO CASE
  CASE BAR() = 1 && Активизировать
    aa =1
    ACTIVATE POPUP F321
  CASE BAR() = 3 && Изменить
    aa =2
    ACTIVATE POPUP F321
  CASE BAR() = 5 && Создать
    ACTIVATE WINDOW Savescr
    listval =SPACE(8)
    @ 10,19 to 12,54
    @ 11,21 say "Введите имя запроса " GET listval
    READ
    CREATE QUERY &listval
    CLEAR GETS
    DEACTIVATE WINDOW Savescr
ENDCASE
SET MESSAGE TO
DEACTIVATE POPUP && Деактивизация вертикального меню P32
RETURN

```

Процедура Pop_F321 позволяет активизировать или изменять структуру файла запроса на выборку данных (с расширением .qbe). Функция PROMPT() возвращает имя файла запроса с маршрутом по директориям, выбранным из списка в окне вертикального меню F321. Это имя присваивается переменной listval, и затем, в зависимости от значения переменной аа (переданного из процедуры Pop_P32), выполняется активизация файла запроса (аа = 1) или выход на его модификацию (аа = 2).

```
PROCEDURE Pop_F321
PARAMETER aa
listval =PROMPT()
DO CASE
  CASE aa = 1 && Активизировать
    SET VIEW TO &listval
  CASE aa = 2 && Модифицировать
    MODIFY VIEW &listval
ENDCASE
SET MESSAGE TO
DEACTIVATE POPUP
RETURN
```

В процедуре Sort, вызываемой из процедуры Pop_P3, с помощью выполнения макроопределения Tabsort, считанного из файла Mac.key, обеспечивается выбор позиции Sort database on field list вертикального меню. Это меню прикреплено к позиции Organize горизонтального меню команды MODIFY STRUCTURE создания/модификации структуры файла БД, что предоставляет пользователю возможность выполнить физическую сортировку файла.

```

PROCEDURE SORT
ACTIVATE WINDOW Savescr
lc_say ='Сортировка записей '
DO info_box WITH lc_say
RESTORE MACROS FROM MAC
PLAY MACRO TABSORT
```

```
MODIFY STRUCTURE
DEACTIVATE WINDOW Savescr
SET MESSAGE TO
RETURN

```

Процедура Pop_P34 реализует функции вертикального меню P34, обеспечивающие переход к первой (команда GO TOP) или последней (команда GO BOTTOM) записи, а также поиск записи по ее физическому номеру (команда GO zn, где zn - номер записи, заданный пользователем).

```
PROCEDURE Pop_P34
DO CASE
  CASE BAR() = 1 && Первая запись
   ACTIVATE WINDOW Savescr
   GO TOP
   DEACTIVATE WINDOW Savescr
  CASE BAR() = 3 && Последняя запись
   ACTIVATE WINDOW Savescr
   GO BOTTOM
   DEACTIVATE WINDOW Savescr
  CASE BAR() = 5 && Запись N
   ZN=0
   @ 13,66 GET zn PICTURE "@Z 99999"
   READ
   GO zn
   CLEAR GETS
ENDCASE
SET MESSAGE TO
DEACTIVATE POPUP
RETURN

```

Процедура Pop_P4 реализует функции вертикального меню P4. В вертикальном меню P4 имеется пять позиций, три из которых (первая "Экранная форма", третья "Отчет", пятая "Марка") активные. Выбор любой из этих позиций активизирует вертикальное меню P41, что приводит к вызову процедуры Pop_P41. Процедуре Pop_P41 из меню P4 передается параметр BB, принимающий значения 1, 2 или 3 в зависимости от выбора одной из активных позиций этого меню. Обычно при передаче в процедуру номера позиции вертикального меню используется функция BAR(). В данном случае из меню P4 передается параметр bb, поскольку функция BAR() в дальнейшем используется для возврата номера позиции меню второго уровня P41, а при реализации функций этого меню требуется использовать выбранное значение позиции меню первого уровня P4 (см. процедуру Printout).

```
PROCEDURE Pop_P4
DO CASE
  CASE BAR() = 1 && Экранная форма
   bb=1
   ACTIVATE POPUP P41
  CASE BAR() = 3 && Отчет
   bb=2
   ACTIVATE POPUP P41
  CASE BAR() = 5 && Марка
   bb=3
   ACTIVATE POPUP P41
ENDCASE
SET MESSAGE TO
DEACTIVATE POPUP
RETURN
```

Процедура Pop_P41 реализует функции вертикального меню P41. К особенностям этой процедуры относится использование вложенных структур команд DO CASE...ENDCASE: после ключевого слова CASE размещается еще одна структура DO CASE...ENDCASE. Вложенная система команд DO CASE... ENDCASE позволяет создать иерархическую систему разветвлений операций. В процедуре Pop_P41 при выборе любой позиции вертикального меню P41 высвечивается вертикальное меню следующего уровня, позиции которого позволяют выполнить функции по разработке форм вывода данных. Например, при выборе позиции "Экранная форма" меню P4 процедуре Pop_P41 передается параметр bb = 1 и выполняется вложенная команда DO CASE... ENDCASE по реализации функций позиций меню P41 "Вызвать" и "Изменить". При реализации функций позиции "Создать" вызывается окно Savescr, в котором осуществляется ввод имени создаваемой экранной формы. Аналогичные операции выполняются и при разработке других форм вывода данных - отчетов и марок.

```

PROCEDURE Pop_P41
PARAMETER bb
DO CASE
  CASE bb = 1 && Экранная форма
   DO CASE
     CASE BAR() = 1 && Подключить
       ACTIVATE POPUP F411
     CASE BAR() = 3 && Создать
       ACTIVATE WINDOW Savescr
       ff =SPACE(8)
       @ 10,15 TO 12,57
       @ 11,17 SAY "Введите имя экранной формы " GET ff
       READ
       CREATE SCREEN &ff
       DEACTIVATE WINDOW Savescr
     CASE BAR() = 5 && Изменить
       ACTIVATE POPUP F413
   ENDCASE
  CASE bb = 2 && Отчет
   DO CASE
     CASE BAR() = 1 && Подключить
       ACTIVATE POPUP F421
     CASE BAR() = 3 && Создать
       ACTIVATE WINDOW Savescr
       rr =SPACE(8)
       @ 10,15 TO 12,49
       @ 11,17 SAY "Введите имя отчета  " GET rr
       READ
       CREATE REPORT &rr
       CLEAR GETS
       DEACTIVATE WINDOW Savescr
     CASE BAR() = 5 && Изменить
       ACTIVATE POPUP F423
   ENDCASE
   CLEAR GETS
  CASE bb = 3 && марка
   DO CASE
     CASE BAR() = 1 && Подключить
       ACTIVATE POPUP F431
     CASE BAR() = 3 && Создать
       ACTIVATE WINDOW Savescr
       ll =SPACE(8)
       @ 10,21 TO 12,56
       @ 11,23 SAY "Введите имя марки  " GET ll
```

```
      READ
      CREATE LABEL &ll
      DEACTIVATE WINDOW Savescr
     CASE BAR() = 5 && Изменить
      ACTIVATE POPUP F433
  ENDCASE
  CLEAR GETS
ENDCASE
SET MESSAGE TO
CLEAR GETS
DEACTIVATE POPUP
RETURN

```

Процедура Pop_P42 реализует функции вертикального меню P42 по выводу предварительно созданного отчета на печать, в текстовый файл или на экран. Вызов процедуры и активизация меню P42 осуществляются после выбора позиции "Вызвать" меню P42 и выбора из высветившегося вертикального меню F421 имени файла отчета. После этого высвечивается вертикальное меню P421 с тремя позициями "На печать", "В текстовый файл" и "На экран".

После выбора позиции "На печать" меню P421 в процедуре Pop_P42 выполняются команды после опции CASE, соответствующей номеру этой позиции. Сначала высвечивается меню для установки параметров печати отчета: размеры страницы, отступ слева, интервал между строками, номер первой печатаемой страницы, тип шрифта и т.д. Эти операции реализуются в процедуре Prt_menu. Процедура Printout осуществляет вывод отчета или марки в зависимости от значения переданного параметра (bb = 2 - вывод отчета, bb = 3 - вывод марки). После выбора позиции для вывода отчета в текстовый файл в окно с именем Tfile следует ввести имя файла и его расширение, а если нужно записать файл на другой диск или директорий, то нужно ввести имя диска и маршрут по директориям (например, C:\DBASE\PLAN\Dok1.txt).

```
PROCEDURE Pop_P42
DO CASE
 CASE BAR() = 3  && Вывод на принтер
  DO Prt_menu
  SET PRINTER ON
  SET CONSOLE OFF
  DO Printout
  SET PRINTER OFF
  SET CONSOLE ON
 CASE BAR() = 5  && Вывод в текстовый файл
  answer = SPACE(8)
  ACTIVATE WINDOW Tfile
  @ 0,0 SAY "—— НАПРАВИТЬ ДОКУМЕНТ В ФАЙЛ ——" COLOR N/G"
  @ 2,1 SAY "Введите имя файла : " GET answer ;
  VALID "" <> TRIM(answer) ;
  MESSAGE "Имя файла состоит не более, чем из 8 символов"
  READ
  DEACTIVATE WINDOW Tfile
  SET CONSOLE OFF
  REPORT FORM &listval TO FILE &answer
  GO TOP
  SET DEVICE TO SCREEN
  SET CONSOLE ON
 CASE BAR() = 7  && Вывод на экран
  CLEAR
  plength  = _plength
  rmargin  = _rmargin
  _plength  = 25
```

```
    _rmargin = 80
    DO Printout
    CLEAR
    _plength = plength
    _rmargin = rmargin
    GO TOP
ENDCASE
DEACTIVATE POPUP
RETURN

```

Процедура Prt_menu осуществляет перед печатью отчета установку параметров страницы и опций печати. Пользователь задает левую и правую границы печати, количество строк на странице, расстояние между строками, с какой страницы начать печать и на какой закончить, число копий и т.д.

```
PROCEDURE Prt_menu
msg_num   = "Введите число"
msg_logic = "Введите Y или N"
msg_enum  = 'Нажмите клавишу "Пробел" для выбора других опций'
loffset  = 0
lmargin  = 0
rmargin  = 80
indent   = 4
plength  = 66
STORE 1 TO pspacing, pbpage, pcopies
pepage   = 9999
peject   = "NONE "
STORE .F. TO pwait, pquality
ppitch   = "PICA   "
ACTIVATE WINDOW Mprint
CLEAR
@ 2, 1 SAY "Параметры страницы"
@ 3, 1 SAY "────────────────────"
@ 4, 1 SAY "Отступ слева       " GET loffset  PICTURE "99"
@ 5, 1 SAY "Левая граница      " GET lmargin  PICTURE "99"
@ 6, 1 SAY "Правая граница     " GET rmargin  PICTURE "999"
@ 7, 1 SAY "Абзац              " GET indent   PICTURE "99"
@ 8, 1 SAY "Длина страницы     " GET plength  PICTURE "99"
@ 9, 1 SAY "Расс. между строк." GET pspacing PICTURE "9" RANGE 1,3
@ 2,26 SAY "Установки печати"
@ 3,26 SAY "────────────────────"
@ 4,26 SAY "Начать печать со стр.N " GET pbpage  PICTURE "999"
@ 5,26 SAY "Закончить на стр. N    " GET pepage  PICTURE "9999"
@ 6,26 SAY "Число копий            " GET pcopies PICTURE "999"
@ 7,26 SAY "Прогон бумаги          " GET peject ;
PICTURE "@M BEFORE,AFTER,BOTH,NONE" MESSAGE msg_enum
@ 8,26 SAY "Ждать между страницами " GET pwait   PICTURE "Y"
@ 9,26 SAY "Тип шрифта             " GET ppitch ;
PICTURE "@M DEFAULT,PICA,ELITE,CONDENSED" MESSAGE msg_enum
@ 10,26 SAY "Качественная печать    " GET pquality PICTURE "Y"
@ 12, 1 SAY "Введите параметры и нажмите клавишу PgDn для сохранения"
READ
DEACTIVATE WINDOW Mprint
_ploffset = loffset
_lmargin  = lmargin
_rmargin  = rmargin
_indent   = indent
_plength  = plength
_pspacing = pspacing
_pbpage   = pbpage
_pepage   = pepage
```

```
_pcopies  = pcopies
_peject   = peject
_pwait    = pwait
_ppitch   = ppitch
_pquality = pquality
RETURN

```

Процедура Printout вызывается из процедуры Pop_P42 и подключает вывод отчета или марки в зависимости от значения переменной bb.

```
PROCEDURE Printout
DO CASE
  CASE bb = 2
    REPORT FORM &listval
  CASE bb = 3
    LABEL FORM &listval
ENDCASE
CLEAR
GO TOP
RETURN

```

Процедура Pop_F411 активизирует вертикальное меню F411, в котором высвечивается список ранее созданных файлов экранных форм (с расширением .fmt). Функция PROMPT() возвращает имя файла экранной формы, выбранного из списка с помощью вертикального меню F411, с указанием диска и маршрута по директориям. Переменной listval присваивается значение функции PROMPT() и затем командой SET FORMAT TO &listval активизируется файл экранной формы.

```
PROCEDURE Pop_F411
listval=PROMPT()
SET FORMAT TO &listval && Подключение файла экранной формы
fr=listval
SET MESSAGE TO
DEACTIVATE POPUP && F411
RETURN

```

Процедура Pop_F413 также активизирует вертикальное меню F413, в котором высвечивается список файлов созданных экранных форм (с расширением .scr). После выбора пользователем требуемого файла экранной формы функция PROMPT() возвращает имя файла с маршрутом по директориям. Переменной listval присваивается значение функции PROMPT() и затем командой MODIFY SCREEN &listval вызывается экран модификации данной формы.

Следующие процедуры с именами Pop_F421, Pop_F423, Pop_F431 и Pop_F433 однотипны по своим функциям. Они вызываются при активизации вертикальных окон, высвечивающих списки файлов отчетов и марок с расширениями соответственно .frg, .frm, .lbg и .lbl.

```
PROCEDURE Pop_F413
listval=PROMPT()
MODIFY SCREEN &listval && Модификация структуры экранной формы
SET MESSAGE TO
DEACTIVATE POPUP && F413
RETURN

PROCEDURE Pop_F421
listval=PROMPT()
ACTIVATE POPUP P42
```

```
SET MESSAGE TO
DEACTIVATE POPUP && F421
RETURN

PROCEDURE Pop_F423
listval=PROMPT()
MODIFY REPORT &listval && Модифиция структуры отчета
SET MESSAGE TO
DEACTIVATE POPUP && F423
RETURN

PROCEDURE Pop_F431
listval=PROMPT()
ACTIVATE POPUP P42
SET MESSAGE TO
DEACTIVATE POPUP && F431
RETURN

PROCEDURE Pop_F433
listval=PROMPT()
MODIFY LABEL &listval && Модифицировать файл марки
SET MESSAGE TO
DEACTIVATE POPUP && F433
RETURN

```

Процедура Pop_P5 реализует функции позиций вертикального меню P5, прикрепленного к позиции "Сервис" горизонтального меню G1. Меню P5 позволяет устанавливать некоторые параметры системы dBASE IV (позиция "Установка параметров dBASE IV"); осуществлять экспорт и импорт файлов (позиции "Экспорт", "Импорт"); выполнять копирование, удаление и переименование файлов (позиции "Копировать", "Удалить", "Переименовать"); вызывать текстовый редактор (позиция "Редактор текста"); вычислять свободную оперативную память (позиция "Память").

При выборе позиции "Редактор текста" вызывается простой и удобный текстовый редактор Norton Editor (версия 3.0, 1987 г.), входящий в состав популярных утилит Питера Нортона (пакет Norton Utilites). Предполагается, что файл вызова редактора ne.com находится на диске C: в директории NU (C:\NU\NE).

Особенностью данной процедуры является использование некоторых команд DOS. Например, для копирования файлов используется команда DOS COPY, которая вызывается командой языка dBASE IV "!": !COPY &co &ck, где co - переменная с именем исходного копируемого файла (файла - источника); ck - переменная с именем выходного файла (файла-мишени), в который будет скопирован файл-источник.

```
PROCEDURE Pop_P5
DO CASE
  CASE BAR() = 1
    ACTIVATE WINDOW Savescr
    SET
    DEACTIVATE WINDOW Savescr
  CASE BAR() = 3 && Экспорт
    ACTIVATE POPUP P52
  CASE BAR() = 4 && Импорт
    ACTIVATE POPUP P53
  CASE BAR() = 6  && Копировать
    ACTIVATE WINDOW Savescr
    co=SPACE(60)
    ck =SPACE(60)
```

```
   @ 8,1 to 10,79 color b/n
   @ 9,2 SAY " Откуда копировать информацию ";
   GET co PICTURE "@S43 "
   @ 11,1 to 13,79 color b/n
   @ 12,2 SAY " Куда копировать               ";
   GET ck PICTURE "@S43 "
   READ
   !COPY &co &ck    && Команда DOS - копирование
   CLEAR GETS
   DEACTIVATE WINDOW Savescr
   CLEAR
  CASE BAR() = 8 && Удалить
   ACTIVATE WINDOW Savescr
   DL =SPACE(60)
   @ 8,2 to 10,78 color b/n
   @ 9,3 SAY " Введите директорий и имя файла для удаления ";
       GET dl PICTURE "@S27 "
   READ
   !DEL &DL && Команда DOS - удаление
   CLEAR GETS
   DEACTIVATE WINDOW Savescr
   CLEAR
  CASE BAR() = 10 && переименовать
   ACTIVATE WINDOW Savescr
   rs =SPACE(60)
   rn =SPACE(60)
   @ 8,1 to 10,79 color b/n
   @ 9,2 SAY  " Какой файл переименовать ? ";
   GET rs PICTURE "@S46 "
   @ 11,1 to 13,79 color b/n
   @ 12,2 SAY " Во что переименовывать  ?  ";
   GET rn PICTURE "@S46 "
   READ
   RENAME &RS TO &RN && Переименовать файл
   CLEAR GETS
   DEACTIVATE WINDOW Savescr
  CASE BAR() = 12 && Редактор текста
   !C:
   !CD C:\NU     && Запуск редактора Norton Editor,
   !NE           && расположенного в директории C:\NU
  CASE BAR() = 14 && Память
   ACTIVATE WINDOW Savescr
   FR ="Свободная оперативная память составляет - ";
   +LTRIM(STR(MEMORY()))+" Кбайт."
   @ 10,14 SAY FR
  SET MESSAGE TO "Для выхода в главн. меню нажмите любую клавишу"
  SET CONSOLE OFF
  WAIT
  SET CONSOLE ON
  DEACTIVATE WINDOW Savescr
ENDCASE
SET MESSAGE TO
DEACTIVATE POPUP && P5
RETURN

```

Процедура Pop_P52 реализует функции вертикального меню P52, которое вызывается после выбора позиции "Экспорт" вертикального меню P5, для обеспечения экспорта активного файла БД в файлы формата PFS, dBASE II, Framework II, RapidFile.

```
PROCEDURE Pop_P52
DO CASE
  CASE BAR() = 1 && PFS-файл
    ACTIVATE WINDOW Savescr
    lc_say='Экспорт записей в файл *.PFS'
    DO info_box WITH lc_say
    tp="PFS"
    DO Proc52 WITH TP
    DEACTIVATE WINDOW Savescr
  CASE BAR() = 3 && dBASEII-файл
    ACTIVATE WINDOW Savescr
    lc_say='Экспорт записей в файл *.dBASEII'
    DO info_box WITH lc_say
    tp="DBASEII"
    DO Proc52 WITH TP
    DEACTIVATE WINDOW Savescr
  CASE BAR() = 5 && FW2-файл
    ACTIVATE WINDOW Savescr
    lc_say='Экспорт записей в файл *.FW2'
    DO info_box WITH lc_say
    tp="FW2"
    DO Proc52 WITH TP
    DEACTIVATE WINDOW Savescr
  CASE BAR() = 7 && RPD-файл
    ACTIVATE WINDOW Savescr
    lc_say='Экспорт записей в файл *.RPD'
    DO info_box WITH lc_say
    tp="RPD"
    DO Proc52 WITH TP
    DEACTIVATE WINDOW Savescr
ENDCASE
SET MESSAGE TO
DEACTIVATE POPUP && P52
RETURN

```

Процедура Proc52 вызывается из процедуры Pop_P52 и выполняет операцию экспорта файла БД в файл формата, указанного в переменной tp, значение которой передается из процедуры Pop_P52.

```
PROCEDURE Proc52
PARAMETER TP
ex=SPACE(8)
@ 10,4 TO 12,76
@ 11,7 SAY " Имя файла, куда будут экспортированы данные " GET EX
READ
SET TALK ON
EXPORT TO &ex TYPE &tp && Экспорт файла
SET TALK OFF
CLEAR GETS
RETURN

```

Процедура Pop_P53 вызывается при активизации вертикального меню P53 и обеспечивает импорт файлов формата PFS, dBASE II, RapidFile, Framework II, Lotus 1-2-3 в файл БД dBASE IV.

```
PROCEDURE Pop_P53
DO CASE
  CASE BAR() = 1 && Формат PFS
    ff="*"
    tp="PFS"
    DO Proc531 WITH ff,tp
  CASE BAR() = 3 && Формат dBASE II
    ff="*.DB2"
```

```
    tp="DBASEII"
    DO Proc531 WITH ff,tp
  CASE BAR() = 5 && Формат RPD
    ff="*.RPD"
    tp="RPD"
    DO Proc531 WITH ff,tp
  CASE BAR() = 7 && Формат Framework II
    ff="*.FW2"
    tp="FW2"
    DO Proc531 WITH ff,tp
  CASE BAR() = 9 && Формат WK1
    ff="*.WK1"
    tp="WK1"
    DO  Proc531 WITH ff,tp
ENDCASE
SET MESSAGE TO
DEACTIVATE POPUP && P53
RETURN
```

Процедура Proc531 вызывается из процедуры Pop_P53 и выполняет операции по определению и активизации вертикального меню F531, высвечивающего список файлов того типа, который передается с переменной ff, для обеспечения операций экспорта или импорта.

```
PROCEDURE Proc531
PARAMETER ff,tp
DEFINE POPUP F531 FROM 10,55 TO 20,70 PROMPT FILES LIKE &FF
ON SELECTION POPUP F531 DO Pop_F531 WITH tp
ACTIVATE POPUP F531
RETURN
```

Процедура Pop_F531 реализует функции вертикального меню F531, которое определяется и активизируется в процедуре Proc531.

```
PROCEDURE Pop_F531
PARAMETER TP
listval = PROMPT()
IF .NOT. "PFS" $ TP
    listval=LEFT('&listval',AT('.','&listval')-1)
ENDIF
ACTIVATE WINDOW Savescr
lc_say='Импорт записей из файла  '+listval
DO info_box WITH lc_say
SET TALK ON
IMPORT FROM &listval TYPE &TP
SET TALK OFF
DEACTIVATE WINDOW Savescr
SET MESSAGE TO
DEACTIVATE POPUP
RETURN
```

Процедура Pop_P6 реализует функции вертикального меню P6, обеспечивающие выход из программы Service в УЦ dBASE IV (командой RETURN TO MASTER) или в DOS (командой QUIT).

```
PROCEDURE Pop_P6
DO CASE
  CASE BAR() = 1 && Выход в dBASE IV
    ACTIVATE WINDOW Savescr
    DEACTIVATE MENU
    DEACTIVATE WINDOW Savescr
```

```
  CASE BAR() = 3 && Выход в DOS
    CLOSE DATABASES
    QUIT
ENDCASE
SET MESSAGE TO
DEACTIVATE POPUP
RETURN
```

Процедура Color устанавливает расцветку различных частей экрана: текст на экране выводится ярко белым цветом на черном фоне, названия позиций меню и сообщений - ярко белым цветом на розовом фоне, заголовки - ярко белым цветом на черном фоне, подсветка курсора - желтым цветом на красном фоне, рамки меню - ярко белым цветом на розовом фоне, текст в строке состояния системы - черным цветом на белом фоне, поля записей файлов БД - ярко голубым цветом на синем фоне.

```
PROCEDURE COLOR
IF ISCOLOR()
  SET COLOR OF NORMAL TO W+/N     && Текст на экране
  SET COLOR OF MESSAGES TO W+/BR  && Меню и подсказки
  SET COLOR OF TITLES TO W+/N     && Заголовки
  SET COLOR OF HIGHLIGHT TO GR+/R && Курсор
  SET COLOR OF BOX TO W+/BR       && Цвет рамки
  SET COLOR OF INFORMATION TO N/W && Строка состоян. системы
  SET COLOR OF FIELDS TO BG+/B    && Поля файла БД
ENDIF
RETURN
```

Процедура Info_Box высвечивает в верхней левой части экрана сообщение о выполняемом действии. Например, при переиндексации файла БД командами ? lc_say и ? REPLICATE ("-",LEN(lc_say)) на экран выводится сообщение "Переиндексация базы данных..." и на следующей строке подчеркивается по всей его длине.

```
PROCEDURE Info_Box
PARAMETERS lc_say
? lc_say
? REPLICATE("-",LEN(lc_say))
?
RETURN
```

Процедура Pause активизируется при возникновении ошибки в программе. Назначение вызова этой процедура осуществляется командой ON ERROR в главной процедуре. Имя программного модуля (процедуры) с номером строки, в которой возникла ошибка, передается в процедуру как параметр. Эти данные записываются в переменную lc_msg. В результате активизируется окно Pause и в нем высвечивается сообщение типа:

Произошла ошибка !!! - Сообщение: File does not exist
Процедура: Service   Строка: 30
Для продолжения нажмите любую клавишу...

```
PROCEDURE Pause
PARAMETER lc_msg
IF TYPE("lc_message")="U"
  gn_error=ERROR()
ENDIF
lc_msg = lc_msg
lc_option='0'
```

```
ACTIVATE WINDOW Pause

IF gn_error > 0
  IF TYPE("lc_message")="U"
    @ 0,1 SAY;
    [Произошла ошибка !!! - Сообщение: ]+MESSAGE()
  ELSE
    @ 0,1 SAY [Ошибка # ]+lc_message
  ENDIF
ENDIF

@ 1,1 SAY lc_msg
DO BELL
WAIT " Для продолжения нажмите любую клавишу..."
DEACTIVATE WINDOW Pause
RETURN

```

Процедура Hlpscr вызывается нажатием клавиши F1 (назначение вызова выполняется в главной процедуре) и позволяет вывести на экран шесть форм справочной информации, которые поясняют функции позиций вертикальных меню первого уровня, прикрепляемых к позициям основного (горизонтального) меню. Формы справочной информации высвечиваются в окне Helpscr. Команда xx = INKEY(0), следующая после команд @...SAY вывода справочной информации, удерживает ее на экране до последующего нажатия пользователем любой клавиши.

```
PROCEDURE Hlpscr
ACTIVATE WINDOW Helpscr
ACTIVATE SCREEN
@ 24,20 SAY "Для продолжения нажмите любую клавишу..."
xx=INKEY()
@ 4,2  SAY "┌──────────────────────────┐" COLOR GR+/G+
@ 5,2  SAY "│ Открыть                  │" COLOR GR+/G+
@ 6,2  SAY "│ Закрыть                  │" COLOR GR+/G+
@ 7,2  SAY "├──────────────────────────┤" COLOR GR+/G+
@ 8,2  SAY "│ Создать                  │" COLOR GR+/G+
@ 9,2  SAY "│ Изменить                 │" COLOR GR+/G+
@ 10,2 SAY "├──────────────────────────┤" COLOR GR+/G+
@ 11,2 SAY "│ Переиндексировать        │" COLOR GR+/G+
@ 12,2 SAY "├──────────────────────────┤" COLOR GR+/G+
@ 13,2 SAY "│ Скопировать              │" COLOR GR+/G+
@ 14,2 SAY "└──────────────────────────┘" COLOR GR+/G+
@ 4,27  SAY "Открыть   - позволяет выбрать БД и открыть ее "
@ 5,27  SAY "                             "
@ 6,27  SAY "Закрыть   - закрывает активный файл БД       "
@ 7,27  SAY "                             "
@ 8,27  SAY "Изменить  - изменяет структуру активной БД   "
@ 9,27  SAY "                             "
@ 10,27 SAY "Переиндексировать - производит переопределение"
@ 11,27 SAY "                   структуры индексного файла"
@ 12,27 SAY "                   в соответствии с изменени-"
@ 13,27 SAY "                   ями в индексных полях     "
@ 14,27 SAY "                             "
@ 15,27 SAY "Скопировать - создает новый  файл БД со структурой"
@ 16,27 SAY "              активного в данный  момент  файла БД"
xx=INKEY(0)
@ 4,2  SAY "┌──────────────────────────┐" COLOR GR+/G+
@ 5,2  SAY "│ Редактировать            │" COLOR GR+/G+
@ 6,2  SAY "├──────────────────────────┤" COLOR GR+/G+
@ 7,2  SAY "│ Добавить                 │" COLOR GR+/G+
@ 8,2  SAY "├──────────────────────────┤" COLOR GR+/G+
@ 9,2  SAY "│ Удалить                  │" COLOR GR+/G+
```

```
@ 10,2  SAY "├──────────────────────────────┤" COLOR GR+/G+
@ 11,2  SAY "│ Просмотреть                  │" COLOR GR+/G+
@ 12,2  SAY "└──────────────────────────────┘" COLOR GR+/G+
@ 13,2  SAY "                                " COLOR B/B
@ 14,2  SAY "                                " COLOR B/B
@ 4,27  SAY "Редактировать - позволяет изменять содержимое   "
@ 5,27  SAY "                записей, высвечиваемых по одной  "
@ 6,27  SAY "                (форма Edit) или в виде списка   "
@ 7,27  SAY "                (Browse таблица)                 "
@ 8,27  SAY "                                                 "
@ 9,27  SAY "Добавить      - добавляет в файл БД новые записи "
@ 10,27 SAY "                                                 "
@ 11,27 SAY "Удалить       - помечает для удаления, удаляет   "
@ 12,27 SAY "                физически или восстанавливает    "
@ 13,27 SAY "                помеченные для удаления записи   "
@ 14,27 SAY "                                                 "
@ 15,27 SAY "Просмотреть  - высвечивает содержимое файла БД   "
@ 16,27 SAY "                                                 "
xx=INKEY(0)
@ 4,2   SAY "┌──────────────────────────────┐" COLOR GR+/G+
@ 5,2   SAY "│ Поиск                        │" COLOR GR+/G+
@ 6,2   SAY "├──────────────────────────────┤" COLOR GR+/G+
@ 7,2   SAY "│ Запрос                       │" COLOR GR+/G+
@ 8,2   SAY "├──────────────────────────────┤" COLOR GR+/G+
@ 9,2   SAY "│ Сортировка                   │" COLOR GR+/G+
@ 10,2  SAY "├──────────────────────────────┤" COLOR GR+/G+
@ 11,2  SAY "│ Запись N                     │" COLOR GR+/G+
@ 12,2  SAY "└──────────────────────────────┘" COLOR GR+/G+
@ 13,2  SAY "                                " COLOR B/B
@ 14,2  SAY "                                " COLOR B/B
@ 4,27  SAY "Поиск       - осуществляет поиск записи      "
@ 5,27  SAY "              по заданному условию           "
@ 6,27  SAY "                                             "
@ 7,27  SAY "Запрос      - позволяет создавать, подключать "
@ 8,27  SAY "              или модифицировать запросы на   "
@ 9,27  SAY "              выборку данных из файлов БД     "
@ 10,27 SAY "                                             "
@ 11,27 SAY "Сортировка  - сортирует записи в возрастающей "
@ 12,27 SAY "              или убывающей последовательности "
@ 13,27 SAY "              по указанным полям              "
@ 14,27 SAY "                                             "
@ 15,27 SAY "Запись N    - переход к первой, последней записи "
@ 16,27 SAY "              или к записи по указанному номеру "
xx=INKEY(0)
@ 4,2   SAY "┌──────────────────────────────┐" COLOR GR+/G+
@ 5,2   SAY "│ Экранная форма               │" COLOR GR+/G+
@ 6,2   SAY "├──────────────────────────────┤" COLOR GR+/G+
@ 7,2   SAY "│ Отчет                        │" COLOR GR+/G+
@ 8,2   SAY "├──────────────────────────────┤" COLOR GR+/G+
@ 9,2   SAY "│ Марка                        │" COLOR GR+/G+
@ 10,2  SAY "└──────────────────────────────┘" COLOR GR+/G+
@ 11,2  SAY "                                " COLOR B/B
@ 12,2  SAY "                                " COLOR B/B
@ 13,2  SAY "                                " COLOR B/B
@ 14,2  SAY "                                " COLOR B/B
@ 4,27  SAY "Экранная форма - предоставляет  возможность   "
@ 5,27  SAY "                 пользователю  разрабатывать,  "
@ 6,27  SAY "                 модифицировать или подлючать  "
@ 7,27  SAY "                 созданные формы для ввода и   "
@ 8,27  SAY "                 вывода данных на экран        "
@ 9,27  SAY "                                               "
@ 10,27 SAY "                                               "
@ 11,27 SAY "Отчет/Марка   - создает или изменяет структуру"
@ 12,27 SAY "                отчета/марки, а также выводит"
@ 13,27 SAY "                на печать, экран или в тексто- "
@ 14,27 SAY "                вый файл сгенерир. отчет/марку "
```

```
@ 15,27 CLEAR TO 18,76
xx=INKEY(0)
@ 4,2  SAY "┌──────────────────────┐" COLOR GR+/G+
@ 5,2  SAY "│ Установки dBASE IV   │" COLOR GR+/G+
@ 6,2  SAY "├──────────────────────┤" COLOR GR+/G+
@ 7,2  SAY "│ Импорт               │" COLOR GR+/G+
@ 8,2  SAY "│ Экспорт              │" COLOR GR+/G+
@ 9,2  SAY "├──────────────────────┤" COLOR GR+/G+
@ 10,2 SAY "│ Копировать           │" COLOR GR+/G+
@ 11,2 SAY "├──────────────────────┤" COLOR GR+/G+
@ 12,2 SAY "│ Удалить              │" COLOR GR+/G+
@ 13,2 SAY "├──────────────────────┤" COLOR GR+/G+
@ 14,2 SAY "│ Переименовать        │" COLOR GR+/G+
@ 15,2 SAY "├──────────────────────┤" COLOR GR+/G+
@ 16,2 SAY "│ Оглавление           │" COLOR GR+/G+
@ 17,2 SAY "├──────────────────────┤" COLOR GR+/G+
@ 18,2 SAY "│ Память               │" COLOR GR+/G+
@ 19,2 SAY "└──────────────────────┘" COLOR GR+/G+
@ 4,27  SAY "Установки dBASE IV - устанавливает параметры  и  "
@ 5,27  SAY "                    опции системы dBASE IV      "
@ 6,27  SAY "                                               "
@ 7,27  SAY "Экспорт - экспортирует файлы БД dBASE IV в файлы "
@ 8,27  SAY "         формата PFS, dBASE II, Framework, RPD  "
@ 9,27  SAY "                                               "
@ 10,27 SAY "Импорт  - импортирует файлы формата PFS, dBASE II"
@ 11,27 SAY "         Framework, RPD, WK1 в файл БД dBASE IV "
@ 12,27 SAY "                                               "
@ 13,27 SAY "Копировать/Удалять  - копирует/удаляет файлы    "
@ 14,27 SAY "                                               "
@ 15,27 SAY "Оглавление - переименовывает выбранный файл в   "
@ 16,27 SAY "            файл с указанным именем             "
@ 17,27 SAY "                                               "
@ 18,27 SAY "Память     - вычисляет свободную  память        "
xx=INKEY(0)
@ 4,2  SAY "┌──────────────────────┐" COLOR GR+/G+
@ 5,2  SAY "│ Control Center       │" COLOR GR+/G+
@ 6,2  SAY "├──────────────────────┤" COLOR GR+/G+
@ 7,2  SAY "│ DOS                  │" COLOR GR+/G+
@ 8,2  SAY "└──────────────────────┘" COLOR GR+/G+
@ 9,2 FILL TO 19,23 COLOR B/B
@ 4,27  SAY "Control Center - завершает работу программы    "
@ 5,27  SAY "                и передает управление  dBASE IV "
@ 6,27  SAY "                                               "
@ 7,27  SAY "                                               "
@ 8,27  SAY "DOS            - завершает работу программы    "
@ 9,27  SAY "                и осуществляет выход в DOS     "
@ 10,27 CLEAR TO 19,76
xx=INKEY(0)
DEACTIVATE WINDOW Helpscr
RETURN
```

Процедура Bell предназначена для звуковой сигнализации о возникшей в ходе выполнения программы ошибке и вызывается из процедуры Pause. При запуске процедуры звучит мелодия, состоящая из трех звуков. Команда SET BELL TO устанавливает частоту и длительность звучания, а команда ?? CHR(7) инициирует звук.

```
PROCEDURE Bell
SET BELL TO 1000,1
?? CHR(7)
SET BELL TO 1300,1
?? CHR(7)
SET BELL TO 1000,3
?? CHR(7)
RETURN
```

ОГЛАВЛЕНИЕ

Научно-технический центр "Эрго"

при издательстве "Радио и связь" выполняет работы по внедрению программно-технических комплексов в области издательской и полиграфической деятельности, созданию новых информационных технологий и автоматизированных рабочих мест в управленческой деятельности.

Незаменимым средством для этого является пакет СУБД dBASE IV, работающий как в индивидуальном, так и сетевом режимах. Настоящая книга служит руководством для овладения этим пакетом пользователями, не имеющими специальной подготовки. Книга полезна и опытным программистам, работающим в среде dBASE.

Все примеры из глав 2 и 5 записаны на дискете, которую можно заказать. На этой дискете также помещается электронный справочник по командам и функциям языка dBASE IV.

ВНИМАНИЕ! Пакет dBASE IV в полном объеме с кириллицей не работает. Для исправления этого недостатка можно использовать русификатор. В продаже имеется такой русификатор для пакета dBASE IV.

Принимаются также заказы на поставку лицензионных пакетов dBASE IV, Framework III и других продуктов фирмы Ashton-Tate.